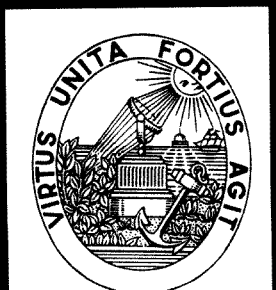
UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA

# Determinação de Indíces de Fiabilidade em Sistemas Eléctricos Utilizando o Método de Monte Carlo

Luís Manuel Martins Vieira Lobo

Porto, Setembro de 2000



**Faculdade de Engenharia da Universidade do Porto**

# Determinação de Índices de Fiabilidade em Sistemas Eléctricos Utilizando o Método de Monte Carlo

Luís Manuel Martins Vieira Lobo

Licenciado em Engenharia Electrotécnica
pela Faculdade de Engenharia da Universidade do Porto

Dissertação submetida para satisfação parcial dos requisitos do grau de mestre
em
Engenharia Electrotécnica e de Computadores



Dissertação realizada sob a supervisão de
Professor Doutor F. Maciel Barbosa
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto, Setembro de 2000

# Resumo

No trabalho realizado foram analisadas algumas noções teóricas relativas à análise de Fiabilidade de Sistemas Eléctricos. Posteriormente, foram desenvolvidos programas computacionais que permitem calcular a Fiabilidade do Sistema Produtor e do Sistema Composto dum Sistema Eléctrico de Energia.

No cálculo da Fiabilidade do Sistema Produtor, usou-se métodos analíticos e o Método de Simulação de Monte Carlo, sendo apresentada uma análise crítica das duas metodologias. Desta forma, é possível estabelecer comparação entre os resultados obtidos através dos dois processos.

No cálculo da Fiabilidade do Sistema Composto, usamos o Método de Simulação de Monte Carlo e o modelo DC para analisar as violações de fluxos nos ramos. Este cálculo é ainda acompanhado pela a visualização de gráficos de probabilidades.

Os programas foram desenvolvidos em linguagem C, dada a grande potencialidade oferecida por esta linguagem, uma vez que permite o uso de apontadores.

## Abstract

In this work, some essential theoretical bases related with the reliability assessment of Electric Power Systems were analysed and described. It were also developed some computational programs that permit the reliability assessment of Generating and Composite Systems of an Electric Power System.

In the reliability assessment of generating system, were used analytical methods and the Monte Carlo Simulation Method. In this way, it was presented a critical analysis of these two processes.

In the reliability assessment of composite system, we used the Monte Carlo simulation method and DC model to analyse flow violations in the lines. This assessment is still accomplished by probabilities graphics visualisation.

The programs were developed in the very powerful C language, because it permits pointers use.

## Resumé

Dans ce travail, ont été analysées et decrivées quelques notions théoriques relatives à l'analyse de la Fiabilité de Systèmes Électriques d'Énergie. Ont été aussi développés quelques programmes pour calculer la Fiabilité du Système Productor et du Système Composé d'un Système Électrique d'Energie.

Quand on fait le calcul de la Fiabilité du Système Productor, nous allons user des méthodes analytiques et la méthode de simulation de Monte Carlo. De cette façon, c'ést présentée une analyse critique des deux méthodes.

Quand on fait le calcul de la fiabilité du système composé, nous allons user la méthode de simulation de Monte Carlo et le modèle DC pour analiser les violations des fluxes à les lignes. Avec ce calcul, il y a aussi une visualisation des graphiques de probabilités.

Les programes ont été developé en language C, une language trés puissante parce-que on peut user des pointeurs.

# Prefácio

Apresento aqui os meus sinceros agradecimentos a todos os que me ajudaram no desenvolvimento desta dissertação, não só pelo auxílio técnico mas também pela motivação que me incutiram. Agradeço nomeadamente:

– Ao meu orientador, o Professor Doutor F. Maciel Barbosa, pela sua disponibilidade, atenção e confiança constantes e, sobretudo, pelo seu rico lado humano;

– Ao Professor Doutor Adriano Silva Carvalho, pela sua disponibilidade e motivação;

– Ao Instituto Superior de Engenharia do Porto pela cooperação e facilidades concedidas;

– Ao meu filho, António, pelo carinho e ajuda no processamento de textos;

– À minha esposa, Amélia, pela compreensão e carinho.

Finalmente, dedico este trabalho, a título póstumo, ao meu pai, Elviro.

# Índice

# Índice de figuras

# Índice de tabelas

# 1. INTRODUÇÃO

## 1.1 – A fiabilidade em sistemas eléctricos

### 1.1.1 – A fiabilidade

O conceito de fiabilidade tornou-se importante a meio da década de 1940 quando a dimensão e a complexidade dos Sistemas Eléctricos começou a aumentar rapidamente. O desenvolvimento de avançados sistemas de armamento originou igualmente estudos de fiabilidade através da simulação em computadores [19]. Esta dissertação vai debruçar-se apenas sobre a Fiabilidade de Sistemas Eléctricos de Energia.

É notório o aumento de interesse em saber os valores esperados da fiabilidade dum Sistema Eléctrico de Energia, de modo a detectar os pontos fracos do sistema e assim evitar avarias e os respectivos custos associados. A maior preocupação é a disponibilidade de serviço, isto é, o tempo durante a qual o serviço é realizado de forma satisfatória. No entanto, a disponibilidade deve ser interpretada com cuidado, pois disponibilidades iguais podem ter efeitos diferentes em utilizadores diferentes. Por exemplo, uma avaria de 10 segundos diários pode ser catastrófica para um computador que necessite de estar sempre ligado (pode haver perda de dados), mas para um utilizador doméstico, pode não ser significativa.

A fiabilidade de um sistema pode ser melhorada utilizando equipamento de maior fiabilidade ou criando circuitos redundantes.

### 1.1.2 – Importância do cálculo de fiabilidade

Melhorar a fiabilidade acarreta custos adicionais devido à instalação de circuitos e equipamentos redundantes. Geralmente, há muitas maneiras de alcançar o mesmo nível de fiabilidade e cada uma delas terá um custo associado. Para seleccionar a solução óptima há que fazer análises técnicas dos custos da avaria e dos custos de cada solução. Mas tudo isto só é possível depois de serem detectados os pontos fracos do sistema, o que

implica um cálculo dos índices de fiabilidade em todos os pontos a alimentar pelo sistema.

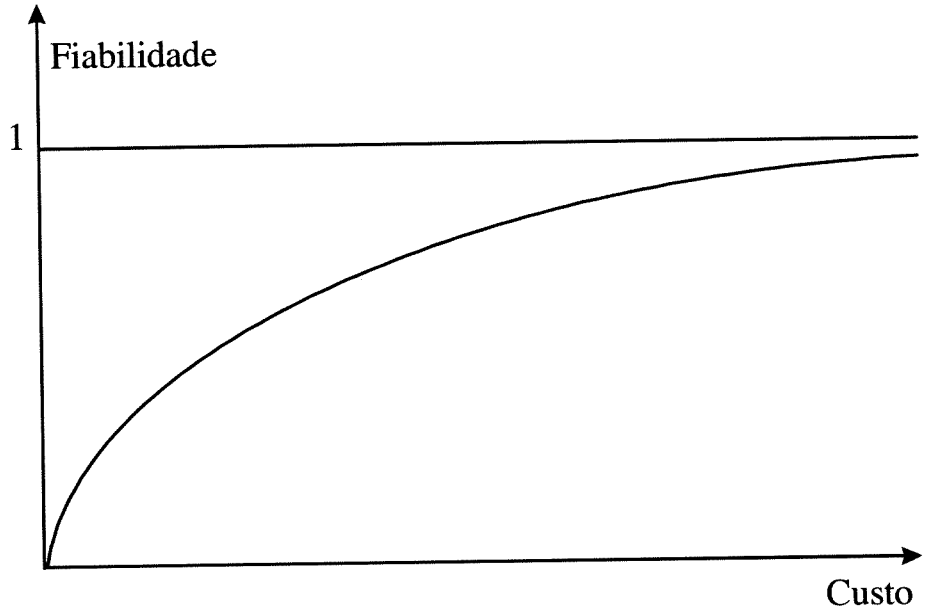



**fig. 1.1 – Custo da fiabilidade de um sistema**

Como se pode ver na fig. 1.1, a fiabilidade de um sistema tende assimptoticamente para 1, pelo que há necessidade de identificar os pontos fracos do sistema para decidir onde é que os investimentos devem ser feitos.

### 1.1.3 – Benefícios do cálculo de fiabilidade

O cálculo de fiabilidade permite aos projectistas do sistema tomarem decisões sobre:

– A escolha da configuração do sistema;

– Fabricante do equipamento;

– Tipo de componentes / equipamento;

– *Interface* com outro equipamento;

– Escolher entre reforçar a fiabilidade e assumir os custos das avarias;

– Escolher o sistema que melhor satisfaz as necessidades dos clientes.

## 1.2 – Estrutura da dissertação

No capítulo 1, como se pode verificar, faz-se uma introdução à Fiabilidade de Sistemas Eléctricos, a qual serve para sensibilizar em várias vertentes para a necessidade da análise de fiabilidade dum sistema. Além disso, também se descreve a estrutura desta dissertação.

No capítulo 2, vamos expõe-se alguns conceitos básicos de fiabilidade. Supõe-se que os componentes estão no período de vida útil, o que implica taxa de avarias constante e que seguem uma distribuição exponencial. Esta distribuição também é devidamente analisada.

No capítulo 3, é descrito o Método de Simulação de Monte Carlo. Critério de convergência, geração de variáveis aleatórias, técnicas de redução da variância, métodos de simulação e modelização da carga num barramento são os assuntos versados.

No capítulo 4, apresenta-se os programas desenvolvidos para calcular a Fiabilidade do Sistema Produtor. São assim apresentados dois programas que usam métodos clássicos mais dois que usam o Método de Monte Carlo e são comparados os resultados obtidos. Será usada a simulação aleatória.

No capítulo 5, apresenta-se um programa para o cálculo dos índices de Fiabilidade do Sistema Composto. Será calculada a fiabilidade em cada barramento do sistema. É usada uma simulação sequencial de forma a se poder calcular os índices de frequência.

No capítulo 6, apresenta-se as conclusões deste trabalho.

Nas referências, aponta-se os livros ou textos nos quais se baseou a presente dissertação, indicando ao longo da mesma chamadas de atenção às referências entre parênteses rectos ([]).

Nos anexos, apresenta-se os programas desenvolvidos codificados em linguagem C.

# 2. CONCEITOS BÁSICOS DE FIABILIDADE

## 2.1 – Taxas de avarias e de reparação de um componente. Probabilidade de um componente estar a funcionar ou avariado

Considere-se que os componentes só podem residir em dois estados: o estado de funcionamento ou o estado de avaria. A fig. 2.1 representa uma sequência de estados de funcionamento e de avaria, bem como a duração de cada estado [6;21;24].

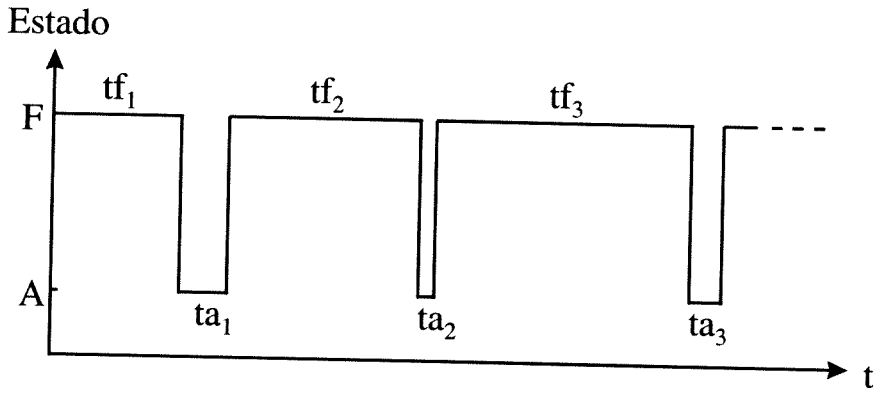



**fig. 2.1 – Sequência de estados dum componente**

A fig. 2.1 representa a história de um componente em que:

– F → estado de funcionamento;

– A → estado de avaria;

– $tf_i$ → tempos de funcionamento (i=1,2,3,...);

– $ta_i$ → tempos de avaria (i=1,2,3,...).

Pode-se então definir os tempos médios de funcionamento m (MTTF – *mean time to failure*) de avaria ou reparação r (MTTR – *mean time to repair*) e o tempo médio entre avarias T (MTBF – *mean time between failure*).

A fig. 2.2 representa graficamente a evolução de funcionamento de um componente.

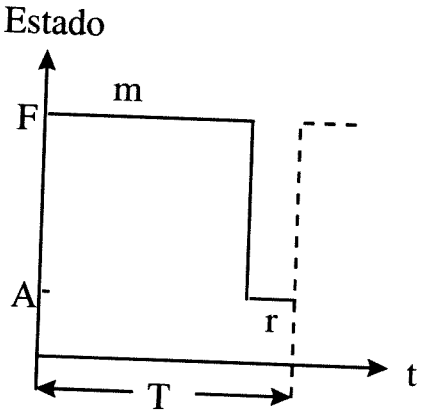



**fig. 2.2 – Tempo médio de funcionamento e de avaria dum componente**

O valor de m é dado por:

$$m = MTTF = \frac{\sum_{i=1}^{nf} tf_i}{nf}$$

onde MTTF significa tempo médio para avariar (*mean time to failure*) e nf é o número de vezes que o componente funcionou.

O valor de r é dado por:

$$r = MTTR = \frac{\sum_{i=1}^{na} ta_i}{na}$$

onde MTTR significa tempo médio para reparar (*mean time to repair*) e na é o número de vezes que o componente avariou.

O valor de T é dado por:

$$T = MTBF = m + r$$

onde MTBF significa tempo médio entre avarias (*mean time between failure*).

O inverso da duração média do período de funcionamento é a taxa de avarias ($\lambda$).

$$\lambda = \frac{1}{m} \qquad (2.1)$$

A taxa de reparação ($\mu$) é o inverso da duração média das avarias.

$$\mu = \frac{1}{r} \qquad (2.2)$$

A probabilidade de o componente estar a funcionar (P(F)) é dada por:

$$P(F) = \frac{m}{m+r} = \frac{\mu}{\lambda+\mu} \qquad (2.3)$$

A probabilidade de o componente estar avariado (P(A)) é dada por:

$$P(A) = \frac{r}{m+r} = \frac{\lambda}{\lambda+\mu} \qquad (2.4)$$

A fig. 2.3 representa a variação da taxa de avaria ao longo do tempo [8;21]. Nesta curva, pode-se considerar três períodos:

– Período I que se designa por "infância". Durante este período, o componente pode apresentar defeitos de fabrico que devem ser corrigidos. Por este facto, a taxa de avaria começa por ser elevada, mas decresce com o tempo à medida que os defeitos de fabrico vão sendo corrigidos;

– Período II, "período de vida útil". Neste período, o componente tem uma taxa de avarias constante;

– Período III, "velhice". Neste período, o componente já está bastante gasto e, como tal, a taxa de avarias aumenta cada vez mais com o tempo.

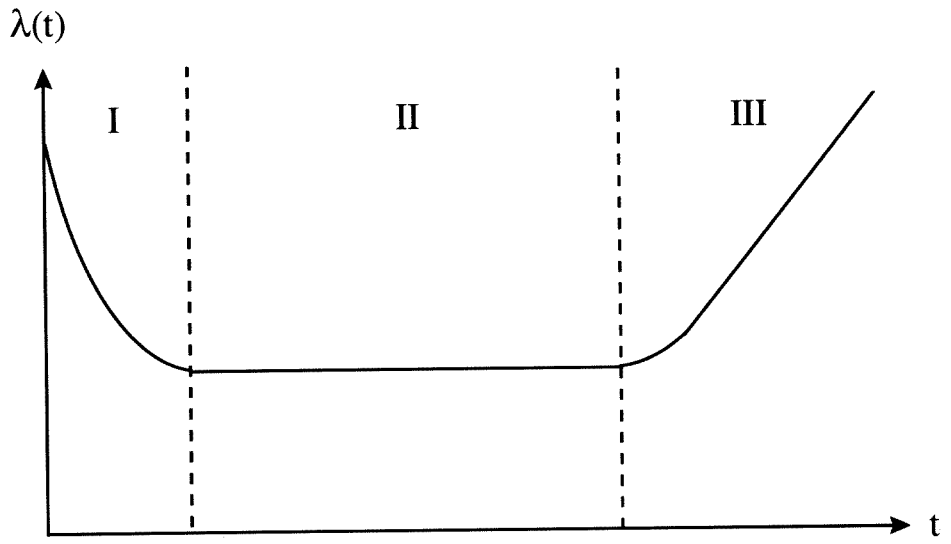



**fig. 2.3 – Variação da taxa de avarias dum componente electrónico**

A situação descrita é a situação típica de um componente electrónico.

Se o componente for electromecânico (ou só mecânico) tem-se na mesma os três períodos focados anteriormente, mas o período de vida útil é bastante reduzido (ver fig. 2.4) [21]. Para aumentar o período de vida útil, recorre-se a acções de manutenção preventiva devidamente programadas.

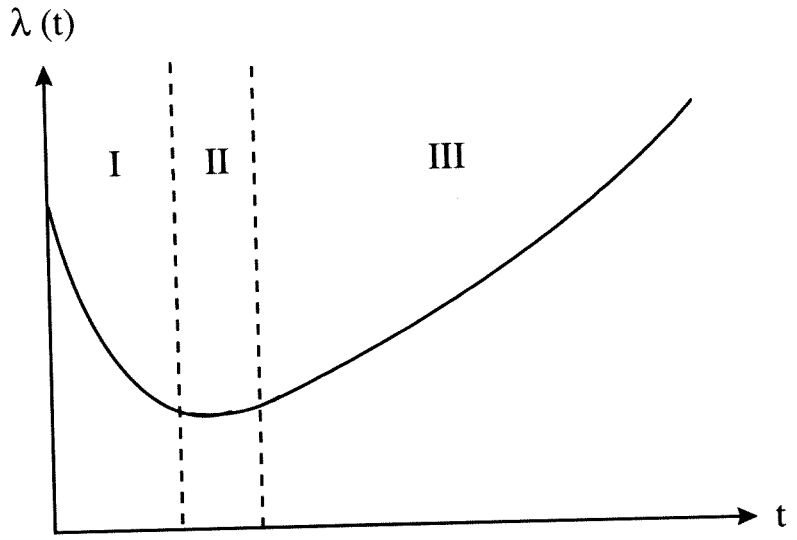



**fig. 2.4 – Variação da taxa de avaria dum componente electromecânico**

Define-se taxa de avarias de um componente no instante t como a probabilidade do componente avariar no intervalo de tempo [t,t+Δt] dividida por Δt. Considerando um universo com N componentes supostos iguais, a taxa de avarias é dada por:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\dfrac{N_a(t+\Delta t) - N_a(t)}{N_s(t)}}{\Delta t} = \frac{1}{N_s(t)} \cdot \frac{d\,N_a(t)}{d\,t} \qquad (2.5)$$

em que:

– $N_s(t)$ representa o número de componentes a funcionar até ao instante t;

– $N_a(t)$ representa o número de componentes avariados até ao instante t;

– $N_a(t+\Delta t)$ representa o número de componentes que avariaram até ao instante $t+\Delta t$;

## 2.2 – Fiabilidade de um componente

"A fiabilidade de um componente é a probabilidade deste desempenhar de forma adequada a função para que foi concebido, nas condições previstas e nos intervalos de tempo em que tal é exigido" [6;21].

Suponha-se que se tem uma amostra de $N_0$ componentes idênticos sem nenhum avariado. Os componentes vão ser todos postos a funcionar a partir de um dado momento, que se considera ser a origem dos tempos. A fiabilidade (R(t)) define-se como sendo a probabilidade de o componente sobreviver até ao instante t, seja:

– $N_s(t)$ o número de componentes da amostra que sobreviveram até ao instante t;

– $N_a(t)$ o número de componentes da amostra que avariaram até ao instante t.

Em qualquer instante verifica-se:

$$N_0 = N_s(t) + N_a(t)$$

A fiabilidade de um componente é dada por:

$$R(t) = \frac{N_s(t)}{N_0} = \frac{N_0 - N_a(t)}{N_0} = 1 - \frac{N_a(t)}{N_0} \qquad (2.6)$$

Derivando a expressão 2.6 em ordem ao tempo, obtém-se:

$$\frac{d\,R(t)}{d\,t} = -\frac{1}{N_0} \cdot \frac{d\,N_a(t)}{d\,t}$$

pelo que:

$$\frac{d\,N_a(t)}{d\,t} = -N_0 \cdot \frac{d\,R(t)}{d\,t}$$

Tendo em consideração a definição de taxa de avarias (expressão 2.5):

$$\frac{d\,R(t)}{d\,t} = -\lambda\,(t)\,.\,R(t)$$

Donde se conclui que:

$$\frac{1}{R(t)}\,d\,R(t) = -\lambda\,(t)\,d\,t \qquad (2.7)$$

Como inicialmente não há componentes avariados, tem-se:

$$t=0 \Rightarrow R(t)=1$$

Integrando a equação 2.7 entre os instantes 0 e t, vem:

$$\int_1^{R(t)} \frac{1}{R(t)}\,d\,R(t) = -\int_0^t \lambda\,(t)\,d\,t$$

ou seja:

$$\ln(R(t)) = -\int_0^t \lambda\,(t)\,d\,t$$

Ou ainda:

$$R(t) = e^{-\int_0^t \lambda\,(t)\,d\,t}$$

Se a taxa de avarias for considerada constante (período de vida útil do componente), a fiabilidade é uma função exponencial:

$$R(t) = e^{-\lambda t} \qquad (2.8)$$

## 2.3 – Probabilidade acumulada de avaria de um componente e respectiva função densidade de probabilidade

Afirmar que um componente avaria até ao instante t é equivalente a afirmar que o componente não sobrevive até esse instante [21]. No período de vida útil, a função de probabilidade acumulada de avaria do componente (Q(t)), que fornece a probabilidade do componente avariar até ao instante t, é dada por:

$$Q(t) = 1 - R(t) = 1 - e^{-\lambda t} \qquad (2.9)$$

A correspondente função densidade de probabilidade é dada por:

$$f(t) = \frac{d\,Q(t)}{d\,t} = \lambda . e^{-\lambda t} \qquad (2.10)$$

Verifica-se assim que durante o período de vida útil, a probabilidade de um componente avariar até ao instante t segue uma distribuição exponencial.

## 2.4 – Valor esperado da distribuição exponencial

O valor esperado da distribuição exponencial é dado por:

$$E(t) = \int_0^{\infty} t . f(t)\, d\,t = \int_0^{\infty} t . \lambda . e^{-\lambda t}\, d\,t$$

Integrando por partes, obtém-se:

$$E(t) = \left[ -t \cdot e^{-\lambda t} \right]_0^{\infty} - \int_0^{\infty} -e^{-\lambda t} \, dt = 0 - \frac{1}{\lambda} \left[ e^{-\lambda t} \right]_0^{\infty} = \frac{1}{\lambda}$$

O valor de E(t) representa o tempo médio para o componente avariar.

$$MTTF = m = E(t) = \frac{1}{\lambda} \qquad (2.11)$$

## 2.5 – Variância e desvio padrão da distribuição exponencial

A variância da distribuição exponencial é dada por:

$$V(t) = E(t^2) - E^2(t) = \int_0^{\infty} t^2 \cdot \lambda \cdot e^{-\lambda t} \, dt - E^2(t)$$

Integrando por partes, tem-se:

$$V(t) = \left[ -t^2 \cdot e^{-\lambda t} \right]_0^{\infty} - \int_0^{\infty} -2t \cdot e^{-\lambda t} \, dt - E^2(t) =$$

$$= 0 + \frac{2}{\lambda} \int_0^{\infty} t \cdot \lambda \cdot e^{-\lambda t} \, dt - E^2(t) = \frac{2}{\lambda} \; E(t) - E^2(t) = \frac{1}{\lambda^2}$$

Em conclusão, tem-se que a variância e o desvio padrão são dados por:

$$V(t) = \sigma^2 = \frac{1}{\lambda^2} \qquad e \qquad \sigma = \frac{1}{\lambda} \qquad (2.12)$$

## 2.6 – Probabilidade de avaria de um componente durante um intervalo de tempo qualquer

Enquanto um componente estiver no período de vida útil ($\lambda$ constante e distribuição exponencial), a probabilidade de avariar durante um dado intervalo de tempo não depende do tempo de funcionamento anterior [21].

Observe-se a fig. 2.5. Considere-se duas origens dos tempos:

– Uma que começa em zero e está inerente á variável t;

– Outra a começar em $t_0$ e está inerente á variável t'.

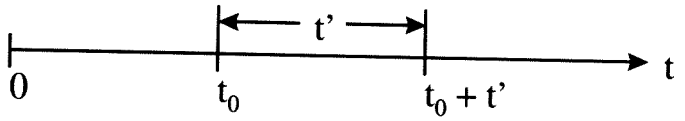



**fig. 2.5 – Origem dos tempos das variáveis t e t'**

Pretende-se provar que a probabilidade de avaria no intervalo $[t_0,t_0+t']$ é igual ao valor da função probabilidade acumulada de avaria com a origem dos tempos a partir de $t_0$, ou seja, igual a Q(t'). O tempo que o componente já funcionou, desde t=0 até t= $t_0$ não vai influenciar a probabilidade de uma futura avaria durante o tempo t'.

Considere-se os seguintes acontecimentos:

– O acontecimento A que consiste em que o componente avarie durante o intervalo de tempo $[t_0,t_0+t']$;

– O acontecimento B que consiste em que o componente sobreviva até ao instante $t_0$.

Tem-se então que provar que a probabilidade de acontecer A, dado que aconteceu B (P(A/B)) é igual a Q(t'). Então:

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \qquad (2.13)$$

$$P(A \cap B) = Q(t_0 \leq t \leq t_0 + t') = Q(t_0 + t') - Q(t_0) =$$

$$= \left[1 - e^{-\lambda(t_0+t')}\right] - \left[1 - e^{-\lambda t_0}\right] =$$

$$= e^{-\lambda t_0} - e^{-\lambda(t_0+t')} \qquad (2.14)$$

$$P(B) = R(t_0) = e^{-\lambda t_0} \qquad (2.15)$$

Substituindo na expressão 2.13 os resultados obtidos em 2.14 e 2.15, obtém-se finalmente:

$$P(A/B) = 1 - e^{-\lambda t'} = Q(t')$$

## 2.7 – Conjunto de estados de um sistema

Admita-se que um sistema é constituído por n componentes e que cada um destes só podem estar num de dois estados possíveis, que são o estado de avaria e o estado de funcionamento.

Um estado do sistema será caracterizado por um vector s [6;24], como se indica a seguir:

$$s = (s_1,\ldots,s_i,\ldots,s_n)$$

em que $s_i$ (i=1,...,n) reflecte o estado do i-ésimo componente do sistema. O conjunto de todos os vectores s referentes a todos os estados possíveis do sistema vai ser designado por G.

Se se considerar que as avarias são acontecimentos aleatórios independentes, a probabilidade de ocorrer um estado s do sistema ($s \in G$) é dada por:

$$P(s) = \left[ \prod_{i \in I} (1 - P_i(A)) \right] . \left[ \prod_{j \in J} P_j(A) \right] \qquad (2.16)$$

em que:

– $P_i(A)$ é a probabilidade de estar avariado o i-ésimo componente do sistema;

– I é o conjunto de índices referentes aos componentes do sistema que estão a funcionar;

– J é o conjunto de índices referentes aos componentes do sistema que estão avariados.

## 2.8 – Valor esperado de uma função associada aos estados de um sistema

Suponha-se que se pretende estudar uma determinada característica de um sistema que, obviamente, é função do estado em que o sistema se encontra. Fica então estabelecida a função F(s) cujo domínio é o conjunto G e cujo valor de F(s) quantifica a característica em estudo. Geralmente, o valor esperado da função F(s) serve como avaliação da referida característica e é dado por:

$$E(F(s)) = E(F) = \sum_{s \in G} F(s) . P(s) \qquad (2.17)$$

Na grande maioria dos casos não é possível utilizar a fórmula 2.17, porque o número de elementos de G é muito elevado. Por exemplo, um sistema com apenas cem componentes e cada componente com dois estados possíveis, daria origem a que G contivesse $2^{100}$ elementos, o que representa $1.27 \times 10^{30}$ estados possíveis do sistema. Como consequência da elevada cardinalidade de G, o valor esperado de F(s) só pode ser estimado mediante uma sondagem ao

conjunto G. É exactamente este o objectivo do Método de Monte Carlo, que vai ser analisado no capítulo 3.

## 2.9 – Conclusões

Neste capítulo dá-se alguns conceitos básicos de fiabilidade necessários à compreensão dos capítulos seguintes. Assim, foram dadas as noções de:

– Taxa de avarias e fiabilidade dum componente;

– Distribuição exponencial;

– Valor esperado e variância da distribuição exponencial;

– Conjunto de estados do sistema;

– Valor esperado duma característica do sistema.

Também foi provado que um componente enquanto se mantiver no período de vida útil não tem memória relativamente à probabilidade de avaria.

# 3. O MÉTODO DE MONTE CARLO

## 3.1 – Introdução histórica

O Método de Monte Carlo [1;4;6;24] data do século XVIII, quando em 1777 o cientista francês Buffon propôs um método para estimar o valor de $\pi$ que consistia no lançamento de uma agulha de comprimento d num plano com linhas paralelas igualmente distanciadas. A distância entre as linhas é a, sendo a>d. A probabilidade da agulha pisar uma linha é $P=2d/(\pi a)$. Após um número significativo de lançamentos estima-se P e sabendo P calcula-se $\pi=2d/(Pa)$.

## 3.2 – Algoritmo genérico do Método de Monte Carlo

Na fig. 3.1, apresenta-se um fluxograma do algoritmo genérico do Método de Monte Carlo [24]. Nmax limita o número máximo de simulações. No caso de não haver convergência, há saída de resultados, mas o utilizador é informado que não houve convergência, o que significa que esses resultados não têm a precisão exigida. No parágrafo seguinte, veremos um critério de convergência baseado na precisão que se pretende obter para os resultados.

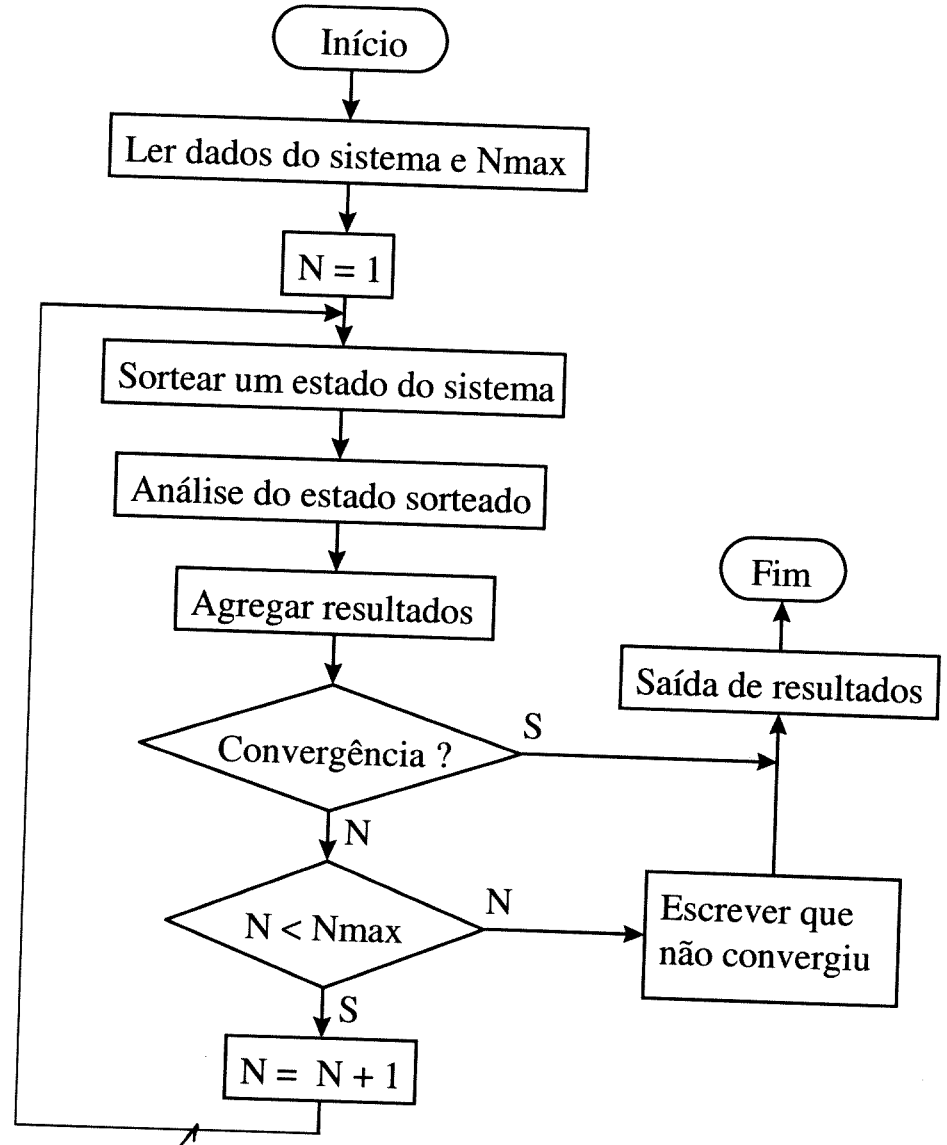



**fig. 3.1 – fluxograma genérico do Método de Monte Carlo**

## 3.3 – Precisão e critério de convergência

Como já foi referido anteriormente, o cálculo do valor exacto de E(F) torna-se impraticável, devido à grande cardinalidade do conjunto de estados do sistema (conjunto G). Portanto, o valor de E(F) vai ser estimado como sendo a média da função F numa amostra de N estados do sistema, ou seja:

$$\bar{E}(F) = \frac{1}{N} \sum_{i=1}^{N} F(s^{(i)}) \qquad (3.1)$$

Um problema que se coloca é saber qual a precisão obtida para E(F) quando se considera a estimativa desse valor [6;24].

Num conjunto de amostras, cada E(F) estimado acaba por ser uma variável aleatória centrada em torno do valor exacto de E(F). A variância de E(F) estimada está relacionada com a variância de F do seguinte modo:

$$V(\bar{E}(F)) = \frac{V(F)}{N} \qquad (3.2)$$

A variância de F pode ser estimada como:

$$\overline{V}(F) = \frac{1}{N-1} \sum_{i=1}^{N} (F(s^{(i)}) - \bar{E}(F))^2 \qquad (3.3)$$

Para N suficientemente grande, a expressão 3.3 pode assumir o aspecto:

$$\overline{V}(F) = \frac{1}{N} \sum_{i=1}^{N} (F(s^{(i)}) - \bar{E}(F))^2 \qquad (3.4)$$

O coeficiente de variação ($\beta$) dá-nos uma ideia da precisão de E(F) estimado e é definido como o quociente entre o desvio padrão de E(F) estimado e o próprio E(F) também estimado, isto é:

$$\beta = \frac{\sqrt{V(\bar{E}(F))}}{\bar{E}(F)} \qquad (3.5)$$

Tendo em conta a expressão 3.2, temos:

$$\beta = \frac{\sqrt{V(F)}}{\sqrt{N}\ \bar{E}(F)} \cong \frac{\sqrt{\overline{V}(F)}}{\sqrt{N}\ \bar{E}(F)} \qquad (3.6)$$

O valor do coeficiente de variação ($\beta$) é muitas vezes usado como critério de convergência do Método de Monte Carlo.

Da expressão 3.6 também se conclui que o aumento da precisão (diminuição do coeficiente de variação) implica a diminuição da variância de F ou o aumento da dimensão da amostra.

É evidente que é desejável que a amostra seja o mais pequena possível. Por isso, no parágrafo 3.6 serão analisadas algumas técnicas de redução da variância.

## 3.4 – Geração de variáveis aleatórias pelo método da transformada inversa

Admitindo que F(x) é uma função monótona crescente cujo contradomínio é o conjunto [0,1] e que U é uma variável aleatória uniformemente distribuída no intervalo [0,1], então a variável aleatória $X=F^{-1}(U)$ tem uma função de distribuição de probabilidade acumulada F(x).

Vamos provar esta afirmação, atendendo à fig. 3.2 [6].

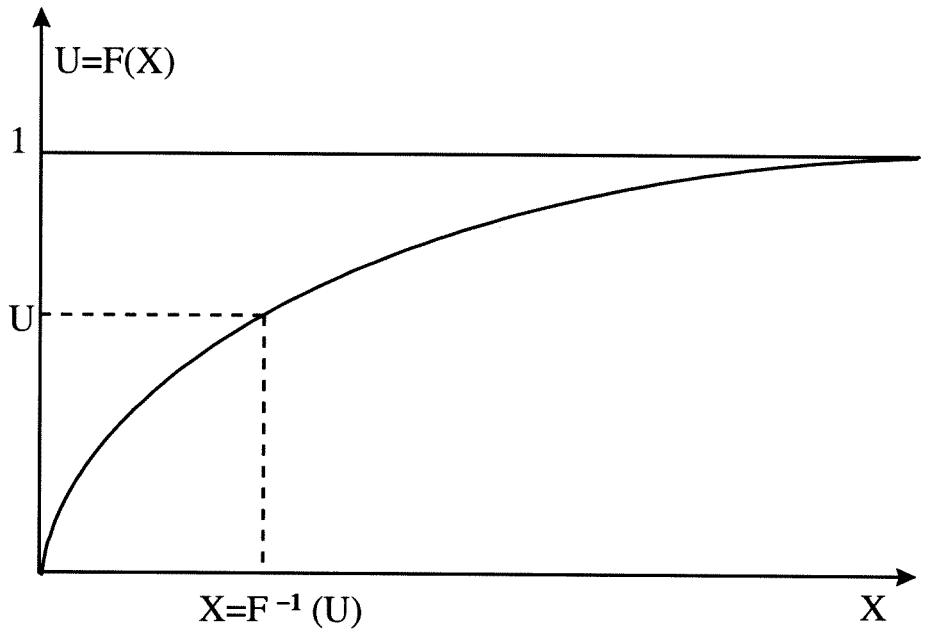



**fig. 3.2 – Exemplificação do método da transformada inversa**

A probabilidade acumulada da variável aleatória X é $P(X \leq x)$. Dado que $X=F^{-1}(U)$, tem-se:

$$P(X \leq x) = P(F^{-1}(U) \leq x)$$

Visto F(x) ser monótona crescente, então:

$$P(F^{-1}(U) \leq x) = P(U \leq F(x))$$

Como U é uma distribuição uniforme, tem-se:

$$P(U \leq F(x)) = F(x)$$

Pelo que:

$$P(X \leq x) = F(x)$$

Conclui-se que a variável aleatória $X=F^{-1}(U)$ segue a função de distribuição F(x). Portanto, para gerar uma variável aleatória que siga essa distribuição é suficiente gerar um número aleatório U segundo uma distribuição uniforme e, seguidamente, calcular a variável aleatória $X=F^{-1}(U)$.

## 3.5 – Geração de variáveis aleatórias exponencialmente distribuídas pelo método da transformada inversa

Como já se viu no parágrafo 2.3, a função de probabilidade acumulada de avaria numa distribuição exponencial [6] (expressão 2.9) é dada por:

$$Q(t) = 1 - e^{-\lambda t}$$

Portanto, temos:

$$U = Q(t) = 1 - e^{-\lambda t}$$

O que dá origem a:

$$X = F^{-1}(U) = -1/\lambda \ln(1 - U) \qquad (3.7)$$

Dado que se U é uniformemente distribuída então também o é 1–U e pode-se simplificar a expressão 3.7 da seguinte forma:

$$X = -1/\lambda \ln(U) \qquad (3.8)$$

## 3.6 – Técnicas de redução da variância

Como já se referiu anteriormente (ver expressão 3.6), a diminuição da variância de F ou o aumento da dimensão da amostra fazem melhorar a precisão com que o valor médio de F é estimado. Portanto, se se reduzir a variância de F, pode-se obter a mesma precisão com uma amostra mais pequena, o que se traduz em menos esforço computacional.

A redução da variância é simplesmente um meio que permite usar no processo de estimação, informação já conhecida.

Nem todas as técnicas teóricas de redução da variância podem ser aplicadas no cálculo de Fiabilidade de Sistemas Eléctricos de Energia.

### 3.6.1 – Redução da variância com uma variável de controlo

Z é uma variável aleatória, cujo valor esperado pode ser obtido por um processo analítico e está fortemente relacionada com a variável aleatória F, cujo valor esperado se pretende estimar [6;24]. Y é outra variável aleatória definida do seguinte modo:

$$Y = F - Z + E(Z)$$

Vê-se então que estimar E(F) é equivalente a estimar E(Y):

$$E(Y) = E(F) - E(Z) + E(Z) = E(F)$$

A variância de Y é, no entanto, menor que a variância de F:

$$V(Y) = V(F) + V(Z) - 2\,cov(F,Z)$$

Como F e Z estão fortemente relacionados, geralmente acontece que:

$$2\,cov(F,Z) > V(Z)$$

Sendo assim, conclui-se que $V(Y)<V(F)$. Portanto, E(F) pode ser calculado mais rapidamente através do cálculo de E(Y). A variável Z é chamada variável de controlo.

### 3.6.2 – Redução da variância considerando as amostras mais importantes

Esta técnica consiste em alterar a função densidade de probabilidade, de modo que passem a ter maior probabilidade de ocorrência os acontecimentos que mais contribuem para os resultados da simulação [13]. A expressão 2.17 dá-nos:

$$E(F) = \sum_{s \in G} F(s) \,.\, P(s)$$

Multiplicando e dividindo por $P^*(s)$ obtemos:

$$E(F) = \sum_{s \in G} \frac{F(s) \,.\, P(s)}{P^*(s)} P^*(s) \qquad \textbf{(3.9)}$$

em que $P^*(s)$ corresponde a uma distribuição que vai contemplar os casos mais importantes para o cálculo de E(F).

Se considerarmos $F^*$ como sendo:

$$F^* = \frac{F(s) \,.\, P(s)}{P^*(s)}$$

então conclui-se da expressão 3.9 que o cálculo de E(F) segundo a distribuição P(s) é igual ao cálculo de $E(F^*)$ segundo a distribuição $P^*(S)$.

A variância de $F^*$ é dada por:

$$V(F^*) = \sum_{s \in G} \left[ \frac{F(s) \,.\, P(s)}{P^*(s)} - E(F) \right]^2 P^*(s)$$

Se se definir $P^*(s)=[F(s).P(s)]/E(F)$, ter-se-ía $V(F^*)=0$, mas tal não é possível uma vez que o valor de E(F) é desconhecido, pois é exactamente o que se pretende calcular. No entanto, se se tiver uma função Z(s) conhecida que é aproximada de F(s), pode-se definir uma aproximação de $P^*(s)$:

$$P'(s) = \frac{Z(s) \,.\, P(s)}{E(Z)}$$

A expressão de cálculo de E(F) assume agora o aspecto:

$$E(F) = \sum_{s \in G} \frac{F(s)}{Z(s)} E(Z)\, P'(s)$$

O valor de E(F) é estimado por:

$$\bar{E}(F) = \frac{1}{N^*} \sum_{i=1}^{N^*} \frac{F(s^{(i)})}{Z(s^{(i)})} E(Z) \qquad (3.10)$$

onde $s^{(i)}$ é o i-ésimo vector de estado sorteado segundo a distribuição de probabilidade P'(s).

### 3.6.3 – Redução da variância usando variáveis antitéticas

Admita-se que é possível encontrar uma variável aleatória Y cujo valor esperado é igual ao da variável aleatória F e que é obtido à custa de outras duas variáveis aleatórias $F_1$ e $F_2$, fortemente relacionadas negativamente [6;24], como a seguir é indicado:

$$Y = \frac{F_1 + F_2}{2}$$

A variância de Y é dada por:

$$V(Y) = \frac{V(F_1) + V(F_2) + 2\mathrm{cov}(F_1,F_2)}{4}$$

Como $F_1$ e $F_2$ estão fortemente relacionados negativamente, então $\mathrm{cov}(F_1, F_2)$ é negativa, o que implica que a variância de Y é inferior àquela que seria obtida caso $F_1$ e $F_2$

fossem independentes. Num processo de amostragem, para se obter $F_1$ e $F_2$ relacionados negativamente, basta proceder do seguinte modo:

– Gerar a sequência $U_1,...,U_n$, que é uma sequência de números aleatórios;

– Obter o estado s do sistema correspondente à sequência anteriormente gerada e considerar $F_1(s)=F(s)$;

– Determinar a sequência $1-U_1,...,1-U_n$ que também é uma sequência de números aleatórios. Esta sequência é designada por sequência antitética da inicial;

– Obter o estado s' do sistema correspondente à sequência antitética e considerar $F_2(s)=F(s')$.

Como $E(F_1)=E(F)$ e $E(F_2)=E(F)$, conclui-se finalmente que:

$$E(Y) = \frac{E(F_1) + E(F_2)}{2} = \frac{E(F) + E(F)}{2} = E(F)$$

## 3.7 – Métodos de simulação

Os métodos de simulação classificam-se consoante o sorteio for de natureza aleatória (não cronológica) ou de natureza sequencial (cronológica). O primeiro método que se descreve é aleatório e os dois últimos são sequenciais.

### 3.7.1 – Simulação através de amostragem aleatória de estados do sistema

Como já se tem vindo a referir, neste método sorteia-se uma amostra de vários estados do sistema, a fim de estimar o valor médio da característica pretendida [6]. Por isso, este método é aleatório, uma vez que os estados são sorteados aleatoriamente sem qualquer sequência cronológica.

Para obter o vector de estado do sistema, há que sortear o estado de cada componente. Assuma-se que cada componente só pode residir num de dois estados possíveis (avaria ou funcionamento) e que as avarias nos componentes são independentes.

O estado do sistema é expresso pelo vector:

$$s = (s_1,\ldots,s_i,\ldots,s_n)$$

em que cada $s_i$ (i=1,...,n) reflecte o estado do i-ésimo componente do sistema do seguinte modo:

$$s_i = \begin{cases} 0 & \text{se } U_i \geq PA_i \quad \text{(estado de sucesso)} \\ 1 & \text{se } 0 \leq U_i < PA_i \quad \text{(estado de avaria)} \end{cases}$$

sendo:

– $PA_i$ a probabilidade de avaria do i-ésimo componente;

– $U_i$ um número aleatório, uniformemente distribuído, pertencente ao intervalo [0,1].

O valor esperado de F (ver expressão 2.17) pode ser estimado do seguinte modo:

$$\bar{E}(F) = \sum_{s \in A} F(s) \frac{n(s)}{N} \qquad \textbf{(3.11)}$$

em que:

– A é o conjunto de estados da amostra;

– n(s) é o número de vezes que ocorreu o estado s na amostra;

– N é a dimensão da amostra.

As vantagens deste método são:

- O sorteio é relativamente simples. Basta somente gerar números aleatórios no intervalo [0,1] uniformemente distribuídos. Não é necessário sortear segundo uma função de distribuição;

- Os dados de fiabilidade exigidos são relativamente poucos. Apenas são necessárias as probabilidades de estado de cada componente;

- A ideia de sortear estados não só se aplica à avaria de componentes mas também pode ser facilmente generalizada ao sorteio de estados para outros parâmetros existentes no cálculo da fiabilidade de sistemas, tais como uma carga, caudais hidrológicos, estados de tempo, etc.

A grande desvantagem deste método é que não pode ser usado para calcular índices de frequência.

### 3.7.2 – Simulação através do sorteio das durações dos estados dos componentes

Este método é baseado no sorteio das durações dos estados dos componentes [6] segundo as respectivas distribuições de probabilidades. Neste método, a sequência de duração dos estados do sistema é obtida pela combinação das sequências previamente sorteadas da duração dos estados dos componentes. Este processo pode repetir-se por vários períodos de tempo de igual amplitude, o que vai permitir estimar índices de fiabilidade inerentes a esse período de tempo (T). A fig. 3.3, mostra um exemplo de como se pode obter, durante o período de tempo T, a sequência da duração dos estados do sistema, supondo que o sistema tem dois componentes.

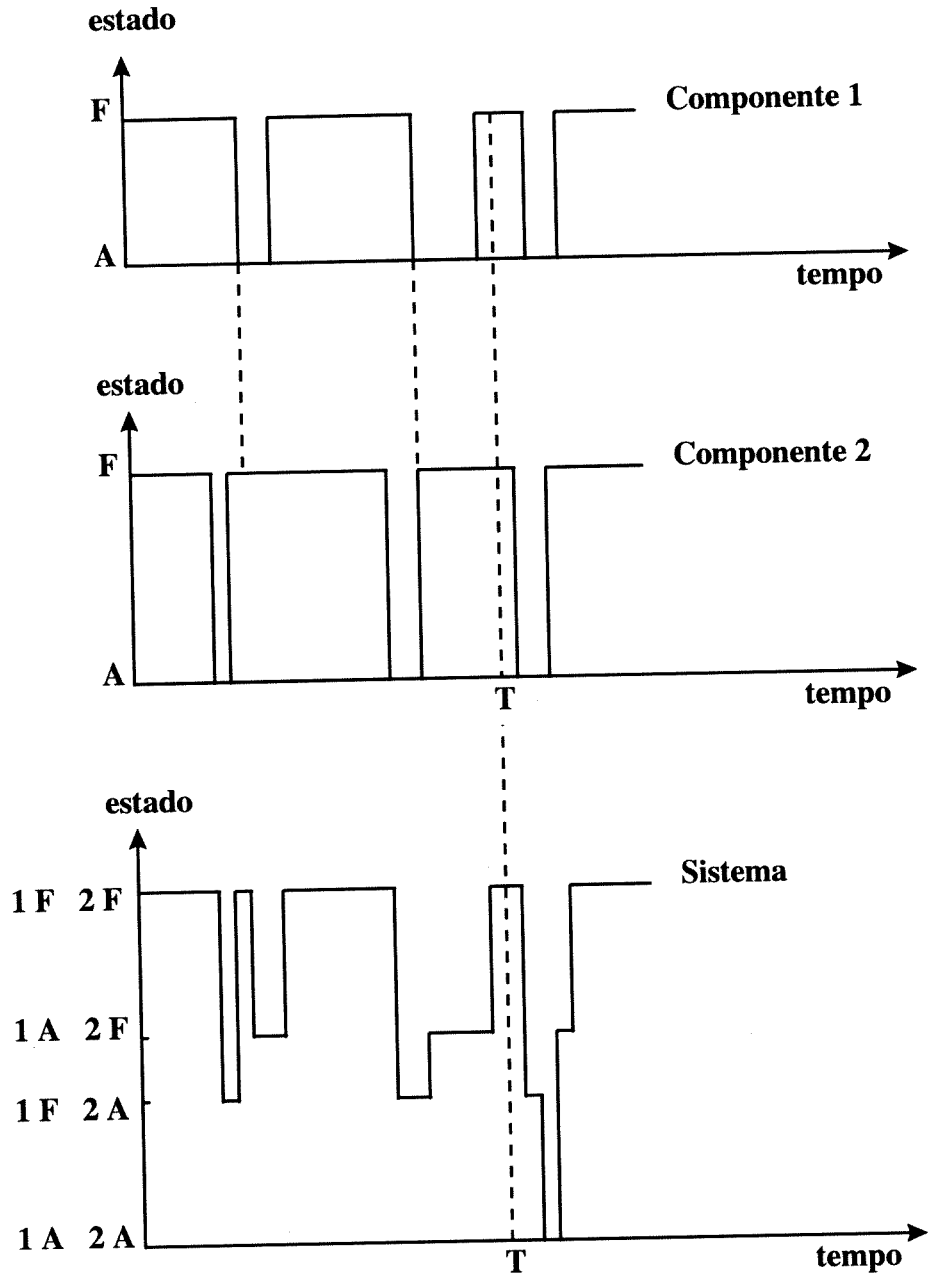



**fig. 3.3 – Exemplificação de como obter os estados do sistema**

Este método pode ser resumido através dos seguintes passos:

– Passo 1: Especificar o estado inicial de cada componente. Geralmente é assumido que todos os componentes estejam inicialmente no estado de funcionamento;

– Passo 2: Para cada componente, sortear a duração de residência no estado actual. Para o caso da distribuição exponencial, usar a fórmula 3.8, ou seja:

$$T_i = - 1/\lambda_i \ln(U_i) \qquad (3.12)$$

– Passo 3: Repetir o passo 2 durante um dado período de tempo (um ano, geralmente) e registar os valores sorteados de cada duração de estado para todos os componentes (ver exemplo da fig. 3.3);

– Passo 4: A evolução cronológica dos estados do sistema pode ser obtida a partir das evoluções cronológicas dos estados de todos os componentes (ver novamente o exemplo da fig. 3.3);

– Passo 5: Analisar cada estado do sistema, de modo a obter os índices de fiabilidade desejados. O uso de vários períodos de tempo permite calcular os valores esperados desses índices.

As vantagens deste método são:

– Pode ser facilmente usado para calcular índices de frequência;

– Qualquer distribuição inerente à duração dos estados pode ser considerada facilmente;

– Além da estimativa dos valores esperados, ainda é possível obter distribuições de probabilidade estatística dos índices de fiabilidade.

As desvantagens deste método são:

– Comparando com a amostragem aleatória de estados do sistema, este método requer mais tempo de computação e mais memória, porque é necessário armazenar as sequências sorteadas de transição de estados de todos os componentes durante um longo período de tempo.

– Este método requer para todos os componentes os parâmetros inerentes a todas as distribuições de duração de estados. Mesmo para componentes com simples distribuições

exponenciais e dois estados já é necessário fornecer duas taxas (taxas de avaria e de reparação) por cada componente. Em alguns casos especiais de componentes com mais de dois estados pode ser difícil obter, num sistema real, todos os dados necessários.

### 3.7.3 – Simulação através do sorteio da transição de estado do sistema

Este método baseia-se no sorteio da transição de um estado do sistema em vez da obtenção das sequências de estados dos componentes [6], como acontecia no método anterior.

Admita-se que o sistema contém m componentes e que as durações de estados desses componentes seguem uma distribuição exponencial. Se, no estado actual do sistema (estado s), $\lambda_i$ (i=1,...,m) forem as taxas de transição de estado de cada componente, então pode-se sortear os tempos de duração dos estados actuais dos componentes ($T_i$ (i=1,...,m)), de acordo com a expressão 3.12. Seguidamente determina-se:

$$T = \min\{T_i : i=1,...,m\} = T_k$$

Pode-se então supor que o sistema vai permanecer no estado actual até que haja um componente que mude de estado, o que é equivalente a afirmar que o sistema vai permanecer no estado s durante o tempo T ($T=T_k$) e que a mudança de estado do sistema de s para outro estado s' é causada pela mudança de estado do componente k. É claro que, no novo estado s', os tempos de duração de estados dos componentes voltam a ser sorteados. Poderia pensar-se que estaria errado aplicar esse procedimento a todos os componentes, uma vez que só o componente k é que transitou de estado. No entanto, se se atender ao exposto no parágrafo 2.6, vê-se que se a duração dum estado dum componente segue uma distribuição exponencial. Então, a probabilidade de o componente avariar em tempo futuro não depende do tempo que o componente já funcionou. Portanto, o método é legítimo, desde que os estados dos componentes sigam distribuições exponenciais.

Para que o sistema transite do estado actual s para o estado s', basta que um componente mude de estado, o que significa

que a taxa de abandono do estado s do sistema é igual ao somatório das taxas de abandono dos estados actuais dos componentes, isto é:

$$\lambda = \sum_{i=1}^{m} \lambda_i \qquad (3.13)$$

Portanto, T segue uma distribuição exponencial cuja função densidade de probabilidade é:

$$f(t) = \lambda\, e^{-\lambda t}$$

Suponha-se que o estado s começa no instante zero e que a transição de s para s' ocorre no instante $t_0$. A probabilidade de esta transição ser causada pela saída do k-ésimo componente do seu estado actual é a seguinte probabilidade condicionada:

$$P_k = P(T_k = t_0 \,/\, T = t_0)$$

De acordo com a definição de probabilidade condicionada tem-se:

$$P_k = \frac{P\left( T_k = t_0 \cap T = t_0 \right)}{P\left( T = t_0 \right)} =$$

$$= \frac{P\left[ T_k = t_0 \cap \left( T_i \geq t_0 \,, i = 1,...,m \right) \right]}{P\left( T = t_0 \right)} =$$

$$= \frac{P\left( T_k = t_0 \right) \prod_{\substack{i=1 \\ i \neq k}}^{m} P\left( T_i \geq t_0 \right)}{P\left( T = t_0 \right)} \qquad (3.14)$$

Dado que $T_i$ (i=1,...,m) e T seguem distribuições exponenciais, tem-se:

$$P(T_i \geq t_0) = \int_{t_0}^{\infty} \lambda_i e^{-\lambda_i t} dt = e^{-\lambda_i t_0} \qquad (3.15)$$

$$P(T_k = t_0) = \lim_{\Delta t \to 0} (\lambda_k e^{-\lambda_k t_0}) \Delta t \qquad (3.16)$$

$$P(T = t_0) = \lim_{\Delta t \to 0} (\lambda e^{-\lambda t_0}) \Delta t \qquad (3.17)$$

Substituindo as equações 3.15, 3.16 e 3.17 na equação 3.14, tem-se:

$$P_k = P(T_k = t_0 / T = t_0) = \frac{\lambda_k}{\lambda}$$

Atendendo à equação 3.13, tem-se finalmente:

$$P_k = \frac{\lambda_k}{\sum_{i=1}^{m} \lambda_i} \qquad (3.18)$$

A transição de estado de qualquer componente do sistema faz com que o sistema transite do estado actual s para outro estado s'. Como o sistema tem m componentes, então há m possibilidades de mudança de estado (ver fig. 3.4). A probabilidade de o sistema mudar para um desses estados é dada pela expressão 3.18.

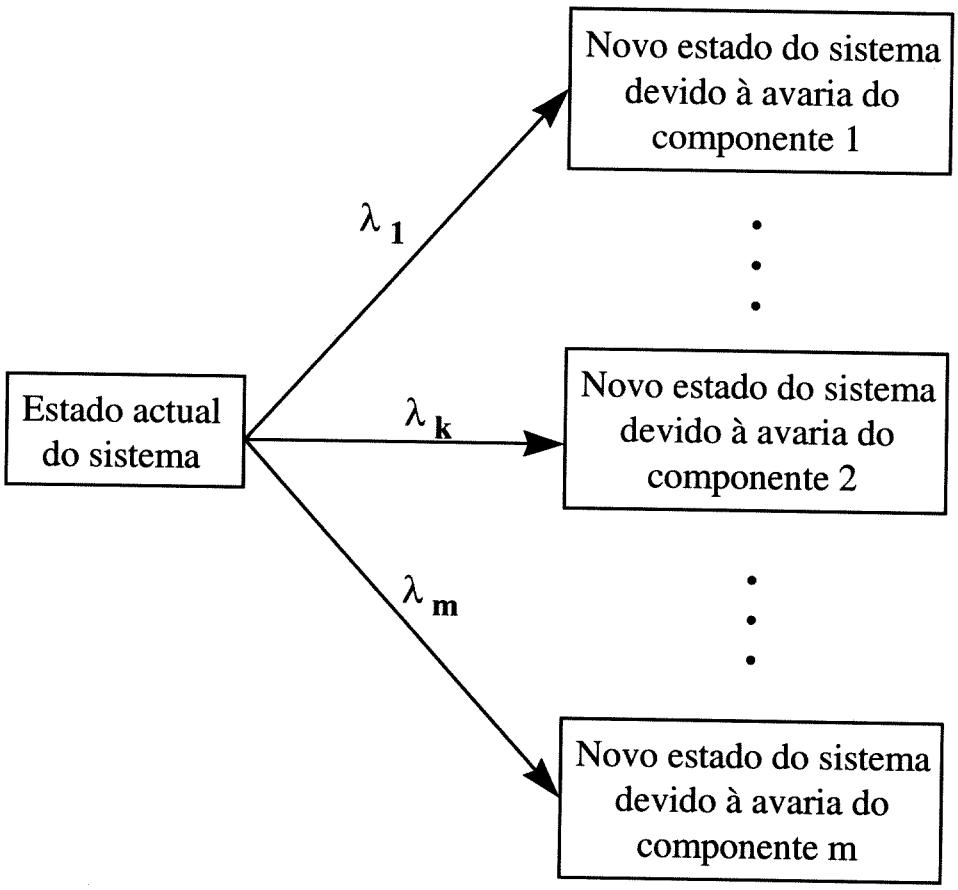



**fig. 3.4 – Transição de estado do sistema**

Note-se que se verifica:

$$\sum_{i=1}^{m} P_i = 1$$

Portanto, o próximo estado pode ser determinado do seguinte modo:

– Colocar sucessivamente no intervalo [0,1] as m probabilidades de saída do estado actual para outro estado, tal como se mostra na fig. 3.5;

– Gerar um número aleatório U, uniformemente distribuído no intervalo [0,1];

– Determinar qual o segmento $P_k$ onde está situado o número aleatório U;

– Considerar que o componente k muda de estado, ficando assim definido o novo estado do sistema.

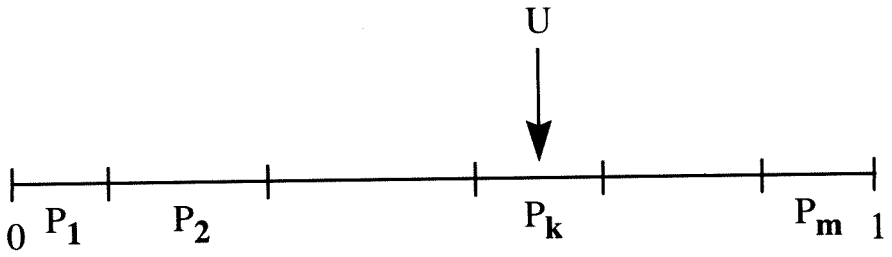



**fig. 3.5 – Sorteio da transição de estado do sistema**

Por este processo, pode-se gerar uma sucessão de estados do sistema e, para cada um deles, efectuar os cálculos necessários, tendo em vista os índices de fiabilidade pretendidos.

As vantagens deste método são:

– Pode ser usado para calcular índices de frequência sem necessidade de sortear funções de distribuição e armazenar informação cronológica, como acontecia na simulação através do sorteio das durações dos estados dos componentes;

– Na simulação através de amostragem aleatória de estados do sistema são necessários m números aleatórios para obter um estado de um sistema que contenha m componentes. Este método requer somente um número aleatório para gerar um estado dum sistema.

Como já se referiu, a desvantagem deste método reside no facto de que só pode ser aplicado quando as durações dos estados dos componentes seguem distribuições exponenciais. Contudo, notemos que as distribuições exponenciais são as mais usadas nos cálculos de fiabilidade.

## 3.8 – Modelização da carga

Vai-se analisar três alternativas de modelizar a carga [6] num determinado barramento do sistema:

a) Considerar o diagrama de cargas cronológico e sortear a carga. Por exemplo, se o diagrama de cargas for anual e nos der as cargas hora a hora, devemos gerar um número inteiro aleatório U uniformemente distribuído no intervalo [1,8760] e a carga sorteada será aquela que corresponde á U-ésima hora do diagrama de cargas.

b) Considerar o diagrama de cargas classificado e aproximá-lo por uma função em degraus, tal como mostramos no exemplo da fig. 3.6.

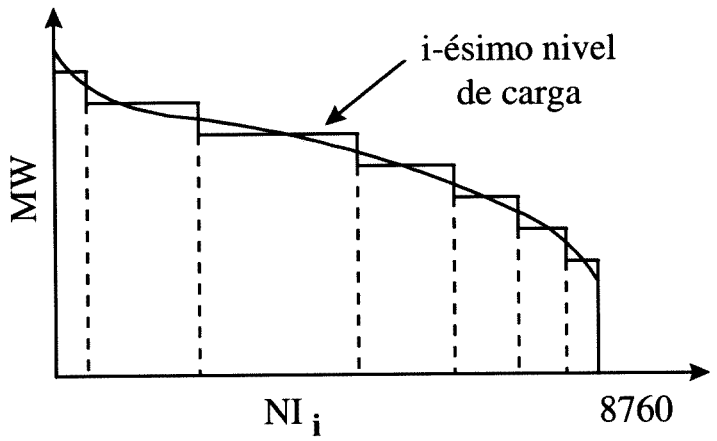



**fig. 3.6 – Aproximação em degrau do diagrama de cargas classificado**

A probabilidade de ocorrer o i-ésimo nível de carga é dada por:

$$P_i = \frac{NI_i}{8760}$$

em que $NI_i$ é a amplitude (número de horas) do i-ésimo intervalo.

Assim, pode-se calcular, para cada nível de carga, o índice de fiabilidade pretendido ($F_i$) e obter o respectivo índice anual (F) através da seguinte média pesada:

$$F = \sum_{i=1}^{NL} F_i \cdot P_i$$

NL é o número de níveis de carga obtidos no diagrama de cargas classificado.

c) Sortear níveis de carga usando uma aproximação por degraus do diagrama de cargas classificado. As probabilidades $P_i$ são colocadas sucessivamente no intervalo [0,1]. Gerar um número aleatório uniformemente distribuído no intervalo [0,1]. O nível de carga é determinado pela localização do número aleatório, tal como se mostra na fig. 3.7.

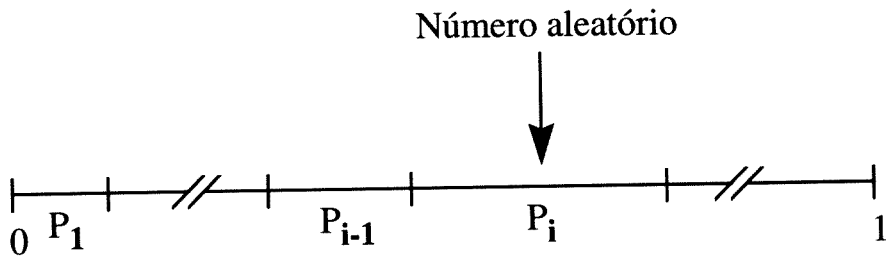



**fig. 3.7 – Sorteio da carga segundo a respectiva probabilidade**

## 3.9 – Conclusões

Neste capítulo, faz-se uma descrição do Método de Simulação de Monte Carlo. Mais especificamente, desenvolvemos os seguintes aspectos relacionados com o Método de Simulação de Monte Carlo:

– Algoritmo genérico;

– Precisão e critério de convergência;

– Geração de variáveis aleatórias, incluindo variáveis aleatórias exponencialmente distribuídas;

– Técnicas de redução da variância;

– Métodos para efectuar a simulação;

– Modelização da carga.

# 4. FIABILIDADE DO SISTEMA PRODUTOR

## 4.1 – Introdução

Para quantificar a Fiabilidade do Sistema Produtor [3;12], vai-se determinar o número de dias no período em estudo em que o sistema de produção não é capaz de alimentar a carga. Esse índice de fiabilidade é designado por LOLE (*Loss of Load Expectation*).

O Sistema Produtor dum Sistema Eléctrico de Energia é composto por geradores com probabilidades de avaria independentes uns dos outros. A probabilidade de avaria de um gerador é designada por FOR (*Forced Outage Rate*) e, segundo a expressão 2.4, é dada por:

$$FOR = \frac{r}{m+r} = \frac{\lambda}{\lambda+\mu}$$

em que:

r – Tempo médio de reparação do gerador;

m – Tempo médio de funcionamento do gerador;

$\lambda$ – Taxa de avaria do gerador;

$\mu$ – Taxa de reparação do gerador.

Sabendo a probabilidade de avaria de cada gerador, é possível construir uma tabela na qual se especifica cada valor de capacidade fora de serviço que pode ocorrer no sistema e a respectiva probabilidade de ocorrência. Repare-se que uma capacidade fora de serviço pode ser atingida por mais do que um estado do sistema. Quando a tabela é demasiado extensa, esta deve ser truncada ou arredondada, o que se irá descrever futuramente.

Uma vez construída a tabela de probabilidades das capacidades fora de serviço e dispondo do diagrama de cargas

classificado, é possível calcular o LOLE através da expressão seguinte:

$$LOLE = \sum_{i=1}^{n} t_i \cdot Prob_i \qquad (4.1)$$

em que:

n – Número de capacidades fora de serviço;

$t_i$ – Tempo durante o qual a carga não é alimentada devido à ocorrência da i-ésima capacidade fora de serviço;

$Prob_i$ – Probabilidade de ocorrer a i-ésima capacidade fora de serviço.

Na fig. 4.1 considera-se a linearização do diagrama de cargas classificado e o significado das variáveis usadas é o seguinte:

ci – Capacidade instalada;

c1 e c2 – Mínimo e máximo do diagrama de cargas classificado;

$cfs_i$ – i-ésimo valor de capacidade fora de serviço;

$cd_i$ – i-ésimo valor de capacidade disponível ($cd_i$=ci–$cfs_i$);

$t_i$ – Tempo durante o qual a capacidade disponível $cd_i$ não consegue alimentar a carga.

Atendendo novamente à fig. 4.1, conclui-se que o valor $t_i$, em percentagem, é dado por:

– Se $cd_i \geq c2$, então $t_i = 0$;

– Se $c1 < cd_i < c2$, então $t_i = (c2-cd) \cdot 100 / (c2-c1)$;

– Se $cd_i \leq c1$, então $t_i = 100$.

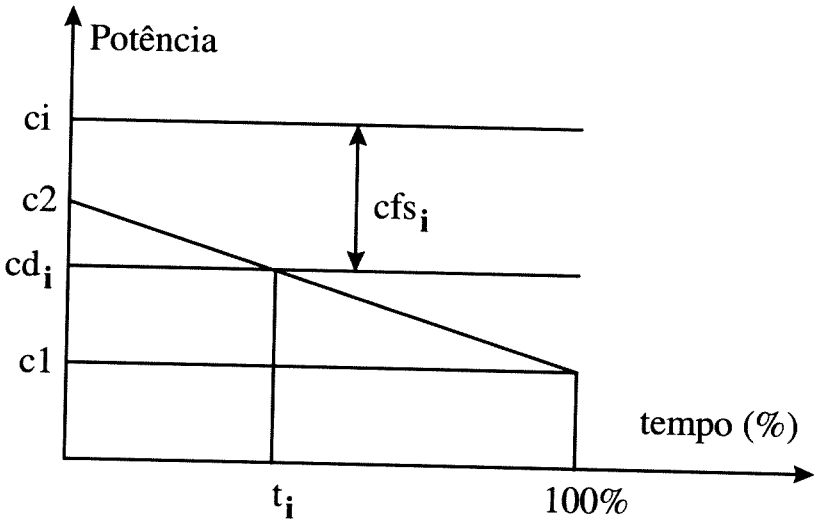



**fig. 4.1 – Exemplificação do cálculo de $t_i$ utilizando a linearização do diagrama de cargas classificado**

## 4.2 – Cálculo do LOLE

Foram desenvolvidos em linguagem C quatro programas para calcular o LOLE. Os programas 1 e 2 utilizam métodos analíticos e os programas 3 e 4 utilizam o Método de Simulação de Monte Carlo.

Os programas 1 e 2 diferem essencialmente no modo como é construída a tabela de probabilidades das capacidades fora de serviço.

O programas 3 dá a evolução gráfica do cálculo do LOLE e o programa 4 calcula o LOLE utilizando como critério de convergência a avaliação do coeficiente de variação ($\beta$).

### 4.2.1 – Sistemas para testar os programas

Para testar os programas desenvolvidos, foram usados os três sistemas seguintes:

– Sistema 1 constituído por cinco geradores de 50MW com FOR = 0.03;

– Sistema 2 constituído por dois geradores de 10MW com FOR = 0.01, um gerador de 15MW com FOR = 0.02 e um gerador de 20MW com FOR = 0.03;

– Sistema 3 ou sistema IEEE [11] para teste de fiabilidade, constituído pelos seguintes geradores:

– cinco de 12MW com FOR = 0.02;

– quatro de 20 MW com FOR = 0.10;

– seis de 50MW com FOR = 0.01;

– quatro de 76MW com FOR = 0.02;

– três de 100MW com FOR = 0.04;

– quatro de 155MW com FOR = 0.04;

– três de 197MW com FOR = 0.05;

– um de 350MW com FOR = 0.08;

– dois de 400MW com FOR = 0.12.

Com estes sistemas, pode-se testar os programas para sistemas apenas com geradores do mesmo tipo (sistema 1), com poucos estados, mas com geradores diferentes (sistema 2) ou com muitos estados e com geradores diferentes (sistema IEEE). Em todos os três sistemas teste, admita-se que o diagrama de cargas classificado é linear, sendo o mínimo e o máximo respectivamente 40% e 80% da capacidade instalada.

Foram comparados os resultados obtidos analiticamente com os obtidos pelo Método de Simulação de Monte Carlo.

### 4.2.2 – Estrutura do ficheiro de entrada de dados

O ficheiro de texto que contém os dados tem a seguinte estrutura:

– Várias linhas, contendo cada uma as seguintes informações correspondentes a um conjunto de geradores do mesmo tipo:

– Potência de cada gerador;

– FOR de cada gerador;

– Número de geradores;

– Uma linha com um zero que indica o local onde terminam as linhas correspondentes aos conjuntos de geradores;

– Outra linha com os valores mínimo e máximo do diagrama de cargas classificado.

### 4.2.3 – Programa 1

#### 4.2.3.1 – Construção da tabela de probabilidades das capacidades fora de serviço

No programa 1, a construção da tabela de probabilidades das capacidades fora de serviço segue os seguintes passos:

1) Construir a tabela correspondente a um só gerador do sistema. Neste caso, a tabela terá só duas capacidades fora de serviço que correspondem aos dois estados possíveis do gerador;

2) Incluir na tabela mais um gerador do sistema. Neste passo, serão desprezadas as capacidades fora de serviço cuja probabilidade de ocorrência seja inferior a um dado valor;

3) Repetir o passo 2 até que tenham sido considerados todos os geradores do sistema.

Através da observação da fig. 4.2, pode-se compreender como se obtém uma nova tabela quando se insere um novo gerador. Numa primeira fase, conjuga-se as capacidades fora de serviço da tabela actual com os dois estados possíveis do novo

gerador, donde advém uma tabela que pode conter capacidades fora de serviço repetidas. Isto significa que podem existir várias alternativas para atingir a mesma capacidade fora de serviço que deverão ser aglutinadas numa só, cuja probabilidade de ocorrência é igual à soma das probabilidades de ocorrência dessas mesmas alternativas. Deste modo e após uma ordenação, obtemos finalmente a nova tabela, na qual já está considerado o novo gerador.

Este método é exacto, mas, infelizmente, só pode ser aplicado a sistemas que se caracterizem por possuir um número de capacidades fora de serviço relativamente pequeno. Para ultrapassar esta dificuldade, trunca-se a tabela, ou seja, despreza-se as capacidades fora de serviço com probabilidade de ocorrência inferior a uma dada probabilidade de truncamento. Mesmo assim, este método ainda continua a ser bastante preciso se a probabilidade de truncamento for baixa, mas, quanto menor for esta probabilidade, maior será a tabela e podem surgir problemas de memória.

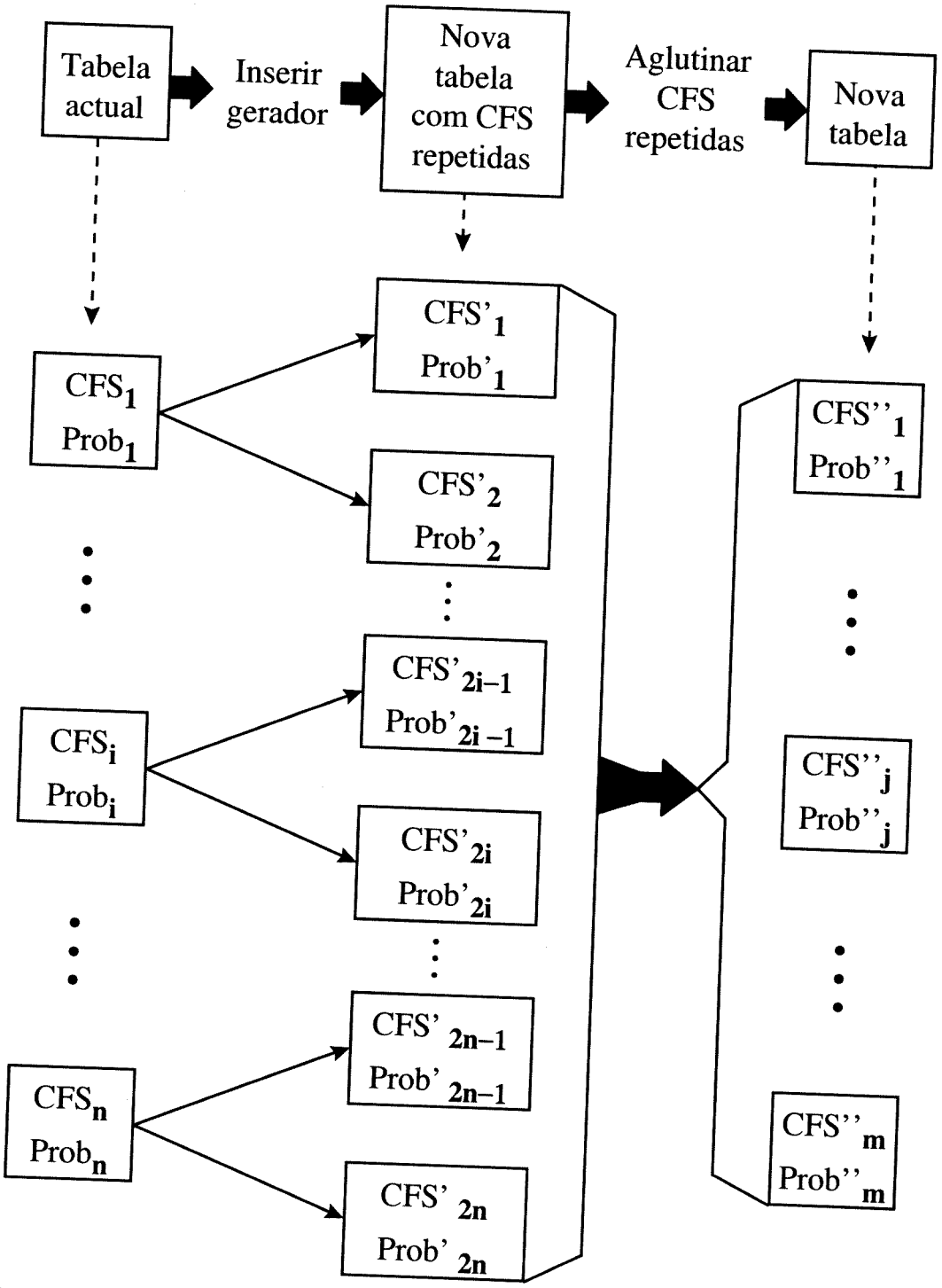



**fig. 4.2 – Processo de construção da tabela de probabilidades das capacidades fora de serviço**

## 4.2.3.2 – Estrutura de dados do programa 1

A seguir, indica-se as principais estruturas de dados usadas no programa 1:

- Vector g onde se armazena as informações relativas aos geradores. Cada elemento desse vector refere-se a um conjunto de geradores do mesmo tipo e contém os seguintes campos:

  - pot → Potência dos geradores (g[i].pot);

  - r → FOR dos geradores (g[i].r);

  - n → Número de geradores do conjunto (g[i].n);

- Variável ng que representa o número de elementos do vector g;

- Vectores p e q onde se armazena as informações relativas a tabelas de probabilidades das capacidades fora de serviço. O vector p refere-se à tabela actual e o vector q refere-se aos dois estados do novo gerador a considerar na tabela actual. Cada elemento destes vectores contém os seguintes campos:

  - cfs → Capacidade fora de serviço (p[i].cfs ou q[j].cfs);

  - prob → Probabilidade de cfs (p[i].prob ou q[j].prob);

- Variáveis np e nq que representam respectivamente o número de elementos dos vectores p e q;

- Variáveis c1 e c2 onde se armazena respectivamente o mínimo e o máximo do diagrama de cargas classificado e linearizado;

- Variável prob_min onde se armazena a probabilidade mínima de truncamento.

### 4.2.3.3 – Fluxogramas do programa 1

Na fig. 4.3, apresenta-se o fluxograma do módulo principal. O programa dá a possibilidade de se poder calcular o LOLE mais do que uma vez com probabilidades de truncamento diferentes. O cálculo da soma das probabilidades desprezadas pretende informar o utilizador do grau de validade dos

resultados. O programa dá a possibilidade de repetir os cálculos com uma probabilidade de truncamento menor, de forma a melhorar a precisão dos resultados, desde que continue a haver memória disponível.

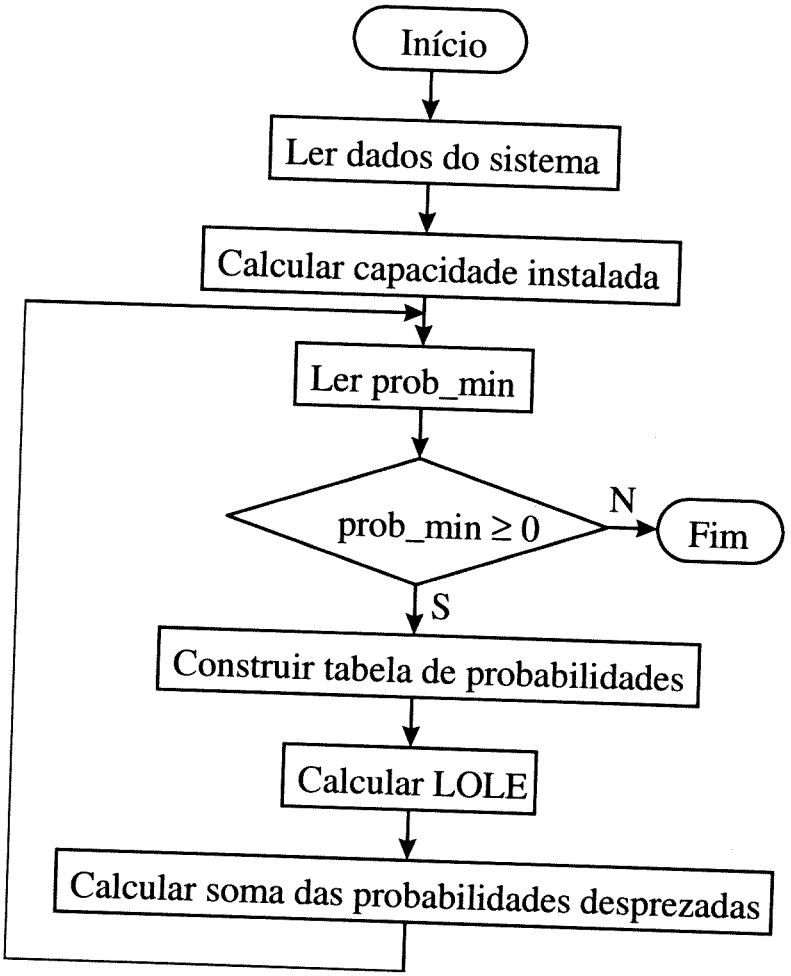



**fig. 4.3 – Módulo principal do cálculo do LOLE com truncamento da tabela de probabilidades das capacidades fora de serviço**

Na fig. 4.4, está o fluxograma do módulo que lê os dados do sistema.

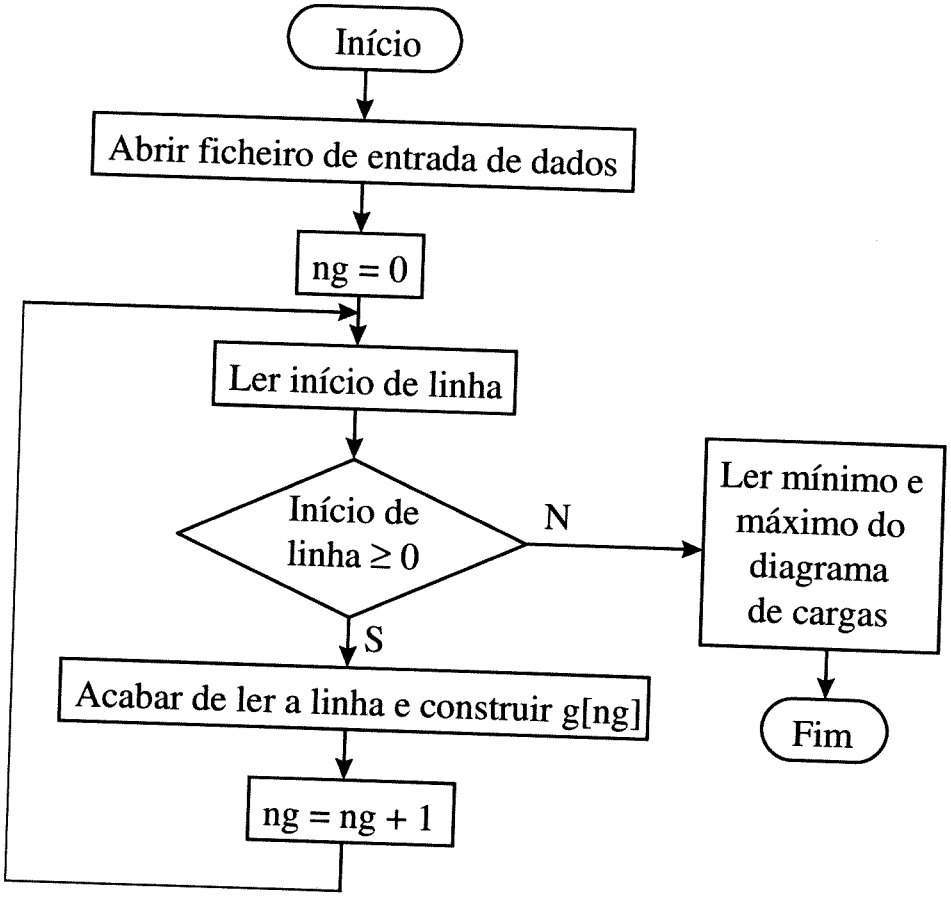



**fig. 4.4 – Módulo de leitura de dados**

Na fig. 4.5, apresenta-se o fluxograma do módulo que calcula a capacidade instalada.

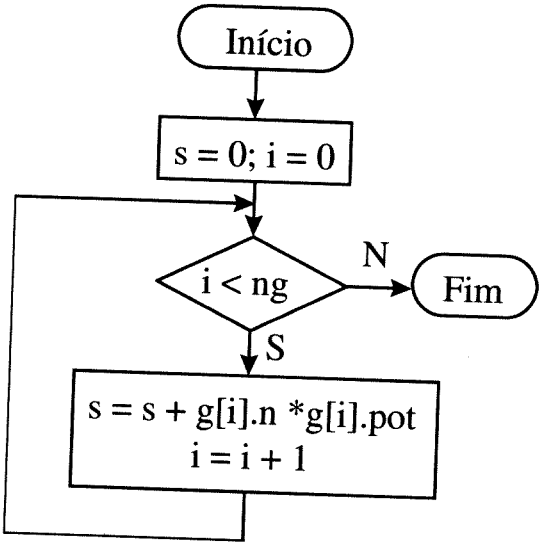



**fig. 4.5 – Módulo do cálculo da capacidade instalada**

Na fig. 4.6, apresenta-se o fluxograma do módulo que constrói a tabela de probabilidades das capacidades fora de serviço.

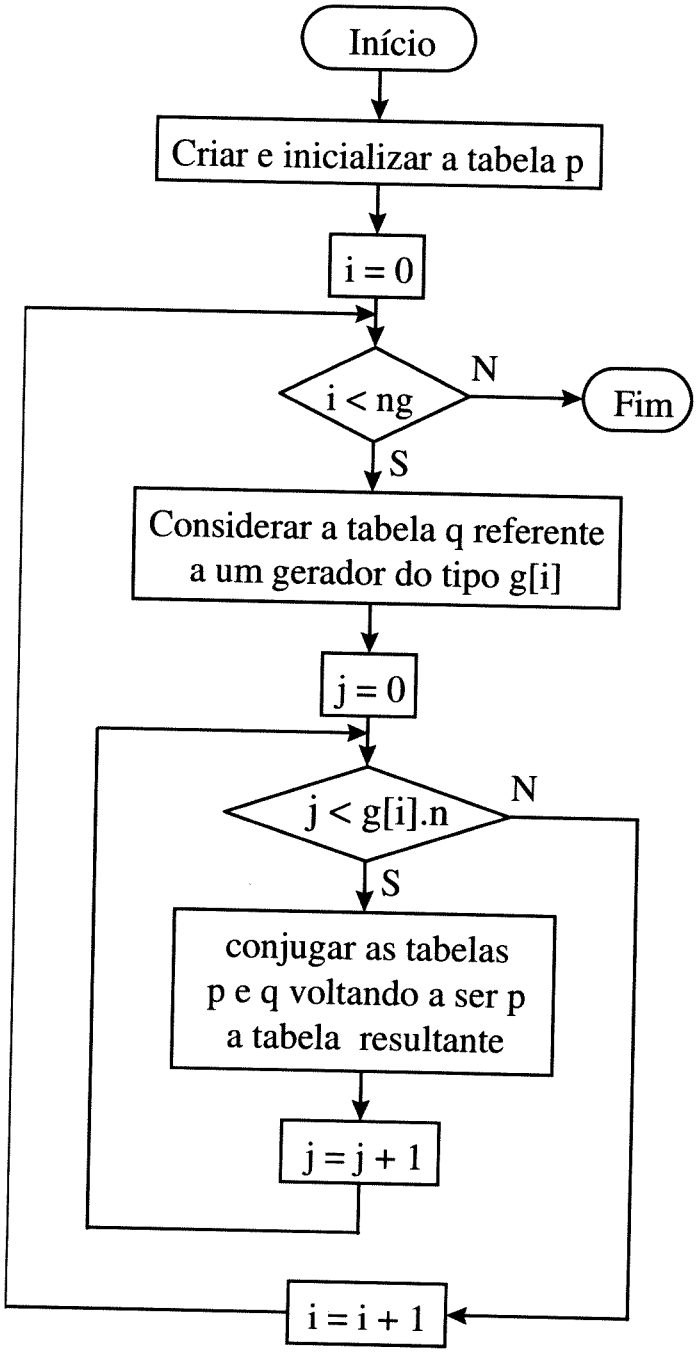



**fig. 4.6 – Módulo de construção da tabela de probabilidades das capacidades fora de serviço**

Nas figs. 4.7 e 4.8, apresenta-se o fluxograma do módulo que conjuga a tabela p com a tabela q. A tabela p é a tabela actual e a tabela q refere-se a um gerador a inserir.

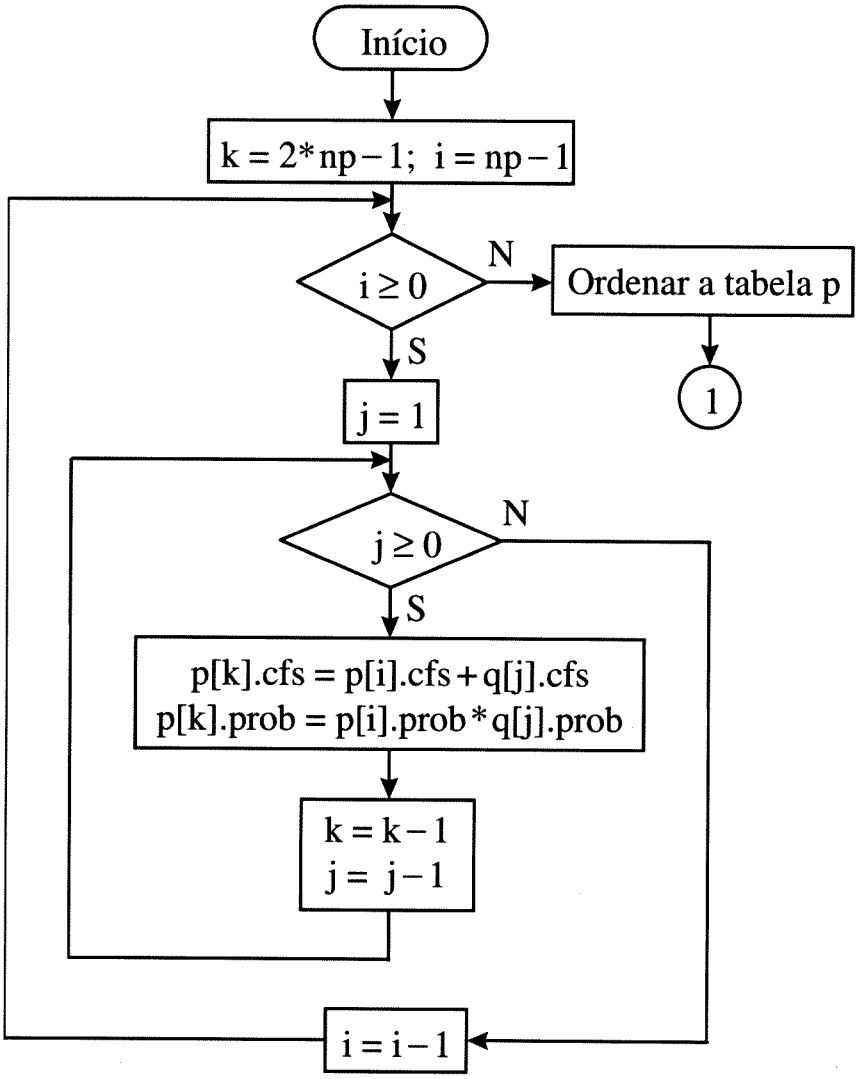



**fig. 4.7 – Módulo que conjuga tabelas de modo a considerar um novo gerador na tabela actual**

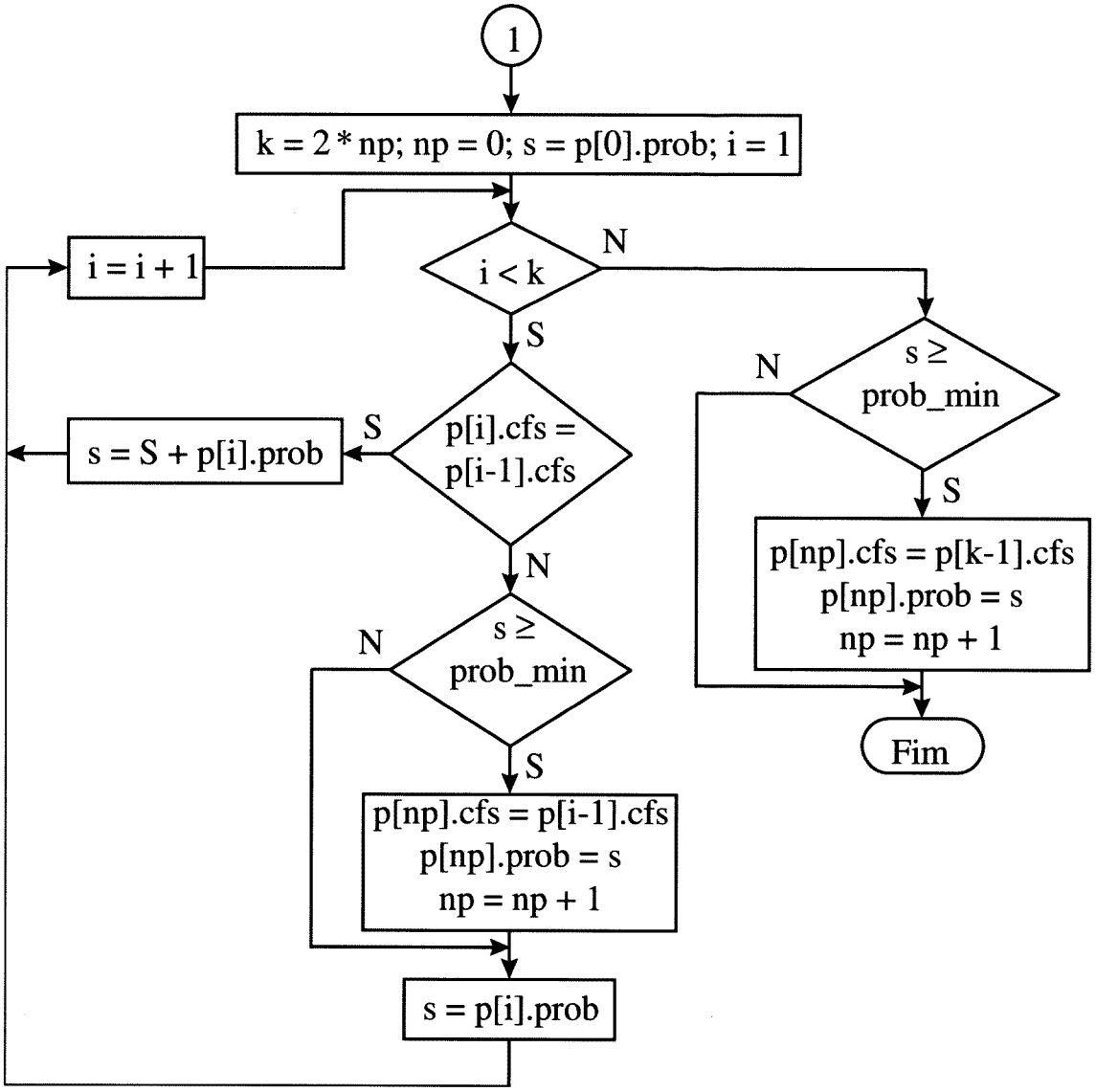



**fig. 4.8 – Continuação do módulo da fig. 4.7**

A fig. 4.9 apresenta o fluxograma do módulo que calcula o LOLE.

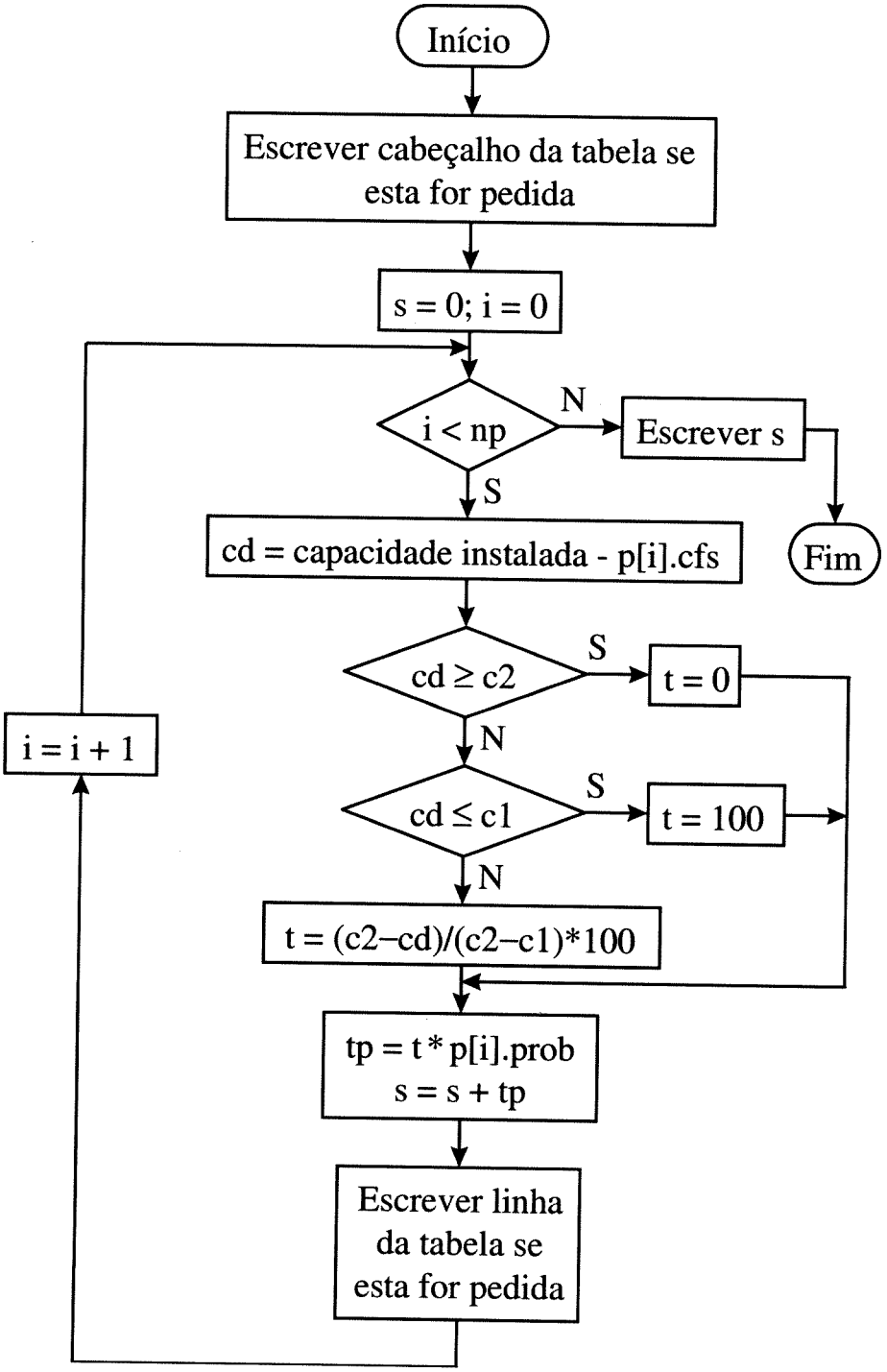



**fig. 4.9 – Módulo do cálculo do LOLE**

Na fig. 4.10, apresentamos o fluxograma do módulo que calcula as probabilidades desprezadas.

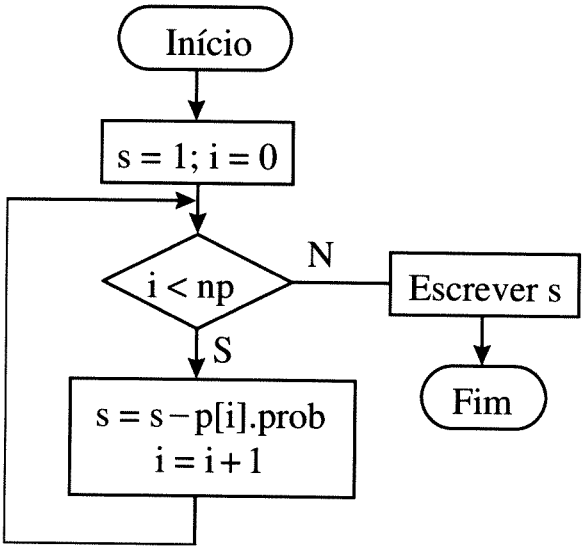



**fig. 4.10 – Módulo do calculo das probabilidades desprezadas**

## 4.2.3.4 – Resultados dos testes do programa 1

Na tabela 4.1, apresenta-se os resultados obtidos pelo programa 1 para os três sistemas de teste considerados anteriormente (parágrafo 4.2.1).

| | Probabilidade de truncamento | LOLE (%) | Probabilidade de desprezada |
|---|---|---|---|
| **Sistema 1** | 0,0E+00 | 0,436502 | 1,0E-16 |
| **Sistema 2** | 0,0E+00 | 1,663596 | 1,1E-16 |
| **Sistema IEEE** | 1,0E-06 | 0,506337 | 5,4E-04 |
| **Sistema IEEE** | 1,0E-07 | Falta memória | Falta memória |

**Tabela 4.1 – Resultados obtidos pelo programa 1**

Note-se que nos sistemas um e dois, a probabilidade de truncamento é zero e a soma das probabilidades desprezadas não é zero. Tal facto acontece devido ao computador ter uma precisão limitada, o que também introduz erros de cálculo. De qualquer forma, pode-se considerar os valores apresentados como os valores exactos (6 decimais) do LOLE dos sistemas um e dois.

Para o sistema IEEE, consegue-se obter resultados com a probabilidade de truncamento igual a $10^{-6}$, mas a tentativa de baixar a probabilidade de truncamento para $10^{-7}$ não foi bem sucedida (memória insuficiente). De qualquer forma, o valor

obtido com a probabilidade de truncamento igual a $10^{-6}$ já é uma boa aproximação do LOLE do sistema IEEE.

Obteve-se ainda as tabelas que permitem o cálculo do LOLE para os sistemas um e dois (tabelas 4.2 e 4.3). No caso do sistema IEEE, não é apresentada tal tabela, dado que esta é demasiado extensa.

| CFS | Prob | T (%) | T * Prob (%) |
|---|---|---|---|
| 0,00 | 0,858734 | 0,00 | 0,000000 |
| 50,00 | 0,132794 | 0,00 | 0,000000 |
| 100,00 | 0,008214 | 50,00 | 0,410703 |
| 150,00 | 0,000254 | 100,00 | 0,025404 |
| 200,00 | 0,000004 | 100,00 | 0,000393 |
| 250,00 | 0,000000 | 100,00 | 0,000002 |
| | 1,000000 | | 0,436502 |

**Tabela 4.2 – Tabela de cálculo do LOLE do sistema 1**

| CFS | Prob | T (%) | T * Prob (%) |
|---|---|---|---|
| 0,00 | 0,931683 | 0,00 | 0,000000 |
| 10,00 | 0,018822 | 0,00 | 0,000000 |
| 15,00 | 0,019014 | 18,18 | 0,345708 |
| 20,00 | 0,028910 | 40,91 | 1,182682 |
| 25,00 | 0,000384 | 63,64 | 0,024444 |
| 30,00 | 0,000582 | 86,36 | 0,050274 |
| 35,00 | 0,000590 | 100,00 | 0,059000 |
| 40,00 | 0,000003 | 100,00 | 0,000294 |
| 45,00 | 0,000012 | 100,00 | 0,001188 |
| 55,00 | 0,000000 | 100,00 | 0,000006 |
| | 1,000000 | | 1,663596 |

**Tabela 4.3 – Tabela de cálculo do LOLE do sistema 2**

## 4.2.4 – Programa 2

### 4.2.4.1 – Construção da tabela de probabilidades das capacidades fora de serviço

Como se viu no programa 1, um sistema com uma tabela de probabilidades das capacidades fora de serviço demasiado extensa pode originar problemas de falta de memória. No programa 2, esse problema vai ser ultrapassado porque a referida tabela vai ser arredondada. Numa tabela arredondada, o número de capacidades fora de serviço é estabelecido

previamente e estas estão igualmente espaçadas. Para cada capacidade fora de serviço que o sistema possuir e que não coincida com nenhuma da tabela arredondada, teremos que distribuir a probabilidade de ocorrência dessa capacidade pelas probabilidades de ocorrência das capacidades fora de serviço mais próximas da tabela arredondada.

Portanto, na tabela arredondada, temos:

$$cfs_i = i \cdot passo \qquad (i = 0,1, \ldots ,n) \qquad \textbf{(4.2)}$$

em que:

- $cfs_i$ → i-ésima capacidade fora de serviço da tabela arredondada;

- passo → Passo das capacidades fora de serviço da tabela arredondada;

- n → Número de elementos da tabela arredondada.

O número n deve ser o primeiro inteiro para o qual se verifica a condição:

$$n \cdot passo \geq \text{capacidade instalada}$$

As probabilidades de ocorrência das capacidades fora de serviço da tabela arredondada são dadas por:

$$P(cfs_i) = \sum_j \left( P(cfs_j') \frac{cfs_j' - cfs_{i-1}}{passo} \right) + \sum_k \left( P(cfs_k') \frac{cfs_{i+1} - cfs_k'}{passo} \right) \qquad \textbf{(4.3)}$$

em que:

- $P(cfs_i)$ → Probabilidade de ocorrência de $cfs_i$;

– $cfs_j$' e $cfs_k$' → j-ésima e k-ésima capacidades fora de serviço da tabela antes de ser arredondada;

– $P(cfs_j')$ e $P(cfs_k')$ → Probabilidades de ocorrência de $cfs_j$' e de $cfs_k$'.

No primeiro somatório, os valores de j são aqueles para os quais se verifica a condição:

$$cfs_{i-1} < cfs_j' < cfs_i$$

No segundo somatório, os valores de k são aqueles para os quais se verifica a condição:

$$cfs_i \leq cfs_k' < cfs_{i+1}$$

Se se usar o sistema P.U. admitindo para potência de base um valor igual ao do passo da tabela, então as fórmulas 4.2 e 4.3 simplificam-se, passando a ter o seguinte aspecto:

$$cfs_i = i \qquad (i = 0,1, \ldots ,n) \qquad \textbf{(4.4)}$$

$$P(cfs_i) = P(i) = \sum_j \left(P(cfs_j') \cdot (cfs_j' - (i-1))\right) + \\ + \sum_k \left(P(cfs_k') \cdot ((i+1) - cfs_k')\right) \qquad \textbf{(4.5)}$$

À tabela com as capacidades fora de serviço em P.U. vai-se chamar tabela normalizada. Em termos de programação, a questão fica simplificada, pois só é necessário um vector para armazenar as probabilidades, uma vez que as capacidades fora de serviço correspondem ao índice desse vector. Além da simplicidade de armazenamento, também se obtém simplicidade ao conjugar tabelas, como veremos a seguir.

Admita-se duas tabelas normalizadas definidas pelos seguintes vectores de probabilidades:

$$P1 \longrightarrow P1(0), \ldots, P1(n1)$$

$$P2 \longrightarrow P2(0), \ldots, P2(n2)$$

Vai-se conjugar as duas tabelas e obter uma tabela definida pelo seguinte vector de probabilidades:

$$P \longrightarrow P(0), \ldots, P(n)$$

Como n é a máxima capacidade fora de serviço da tabela que resulta da conjugação das duas tabelas iniciais, então tem-se:

$$n = n1 + n2$$

Por outro lado tem-se:

$$P(k) = \sum_{i} \sum_{j} P1(i) \cdot P2(j) \qquad \textbf{(4.6)}$$

em que:

$k = 0, \ldots, n;$

$i + j = k;$

$i \in \{0, \ldots, n1\};$

$j \in \{0, \ldots, n2\}.$

Note-se que i + j = k assegura que só são somadas as probabilidades alternativas referentes à capacidade fora de serviço k da tabela final.

A fórmula 4.6 ainda se pode simplificar se se atender ao gráfico da fig. 4.11. Todas as combinações de i e j situam-se no rectângulo assinalado e pertencem à recta i + j = k. Assim sendo, para um dado k, é sempre possível definir os valores de

início (i1) e de fim (i2) da variação do i, o que permite simplificar a fórmula 4.6, como se indica a seguir:

$$P(k) = \sum_{i=i1}^{i2} P1(i) \,.\, P2(k-i) \qquad (4.7)$$

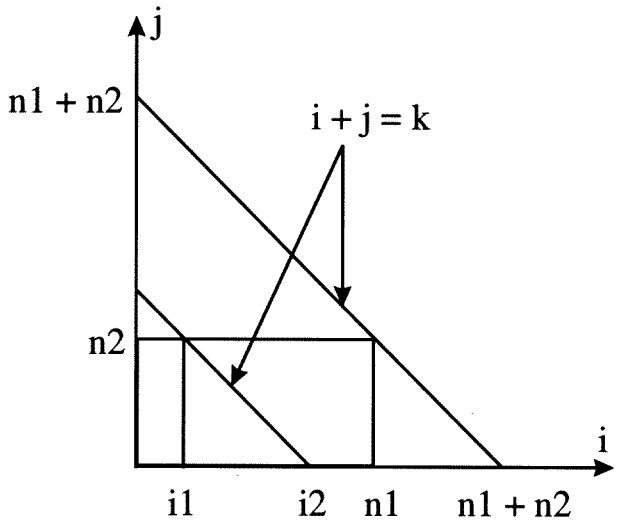



**fig. 4.11 – Exemplificação geométrica de como obter os pares ordenados (i,j) para aplicação na fórmula 4.6**

Para calcular i1 e i2 atende-se novamente à fig. 4.11 e conclui-se que:

$$i1 + n2 = k \Leftrightarrow i1 = k - n2$$

$$i2 + 0 = k \Leftrightarrow i2 = k$$

Se o valor calculado de i1 for negativo, então i1 deverá ser considerado zero. Se o valor calculado de i2 for maior que n1, então i2 deverá ser considerado n1.

Em conclusão:

$$i1 = \begin{cases} k - n2 & \text{se} \quad k > n2 \\ 0 & \text{se} \quad k \leq n2 \end{cases} \qquad (4.8)$$

$$i2 = \begin{cases} k & \text{se} \quad k < n1 \\ n1 & \text{se} \quad k \geq n2 \end{cases} \qquad \textbf{(4.9)}$$

Este método ainda tem um pequeno problema que será esclarecido a seguir através de um exemplo.

Considere-se um sistema com 1,3 P.U. de potência instalada e outro com 5,2 P.U. de potência instalada, o que implica que as respectivas tabelas normalizadas tenham as capacidades fora de serviço a variar de 0 P.U. a 2 P.U. e de 0 P.U. a 6 P.U.. Ao fazer a conjugação das duas tabelas obter-se-ía uma tabela cujas capacidades fora de serviço variavam de 0 P.U. a 8 P.U.. Mas a potência instalada dos dois sistemas só totalizaria 6,5 P.U., o que significa que, na tabela resultante, as capacidades fora de serviço só deveriam ir até 7 P.U.. Sendo assim, não é descabido considerar que a probabilidade de ocorrência da capacidade fora de serviço igual a 8 P.U. se acumula com a de 7 P.U..

Finalmente, tem-se que, no programa 2, a construção da tabela de probabilidades das capacidades fora de serviço segue os seguintes passos:

1) Considerar inicialmente a tabela normalizada correspondente a um conjunto de geradores iguais do sistema. Para obter esta tabela, usa-se a distribuição binomial, isto é:

$$P_k = C_k^n (FOR)^k (1 - FOR)^{n-k}$$

em que:

– n → Número de geradores;

– k → Número de geradores avariados;

– FOR → *Forced Outage Rate* de cada gerador;

– $P_k$ → Probabilidade de estarem k geradores avariados;

$$C_k^n = \frac{n!}{k! \,.\, (n-k)!}$$

2) Considerar mais um conjunto de geradores iguais do sistema e fazer a respectiva modificação na tabela. Neste passo, serão conjugadas duas tabelas normalizadas, tal como descrevemos anteriormente.

3) Repetir o passo 2, até que tenham sido considerados todos os geradores do sistema.

Neste método, em que se faz um arredondamento de tabelas, nenhuma probabilidade vai ser desprezada, mas o próprio arredondamento introduz erro.

### 4.2.4.2 – Estrutura de dados do programa 2

As principais estruturas de dados são as seguintes:

– Vector g igual ao que foi definido no parágrafo 4.2.3.2;

– Variável ng que representa o número de elementos do vector g;

– Vectores p e q onde se armazena as informações relativas à tabela de probabilidades das capacidades fora de serviço. Cada elemento destes vectores armazena a probabilidade de ocorrer uma capacidade fora de serviço igual ao índice desse mesmo elemento. O vector p refere-se à tabela actual e o vector q refere-se a um conjunto de geradores iguais, que se pretende considerar na tabela actual;

– Variáveis fp e fq que representam respectivamente o índice final dos vectores p e q;

– Variáveis c1 e c2 onde se armazena respectivamente o mínimo e o máximo do diagrama de cargas classificado;

– Variável fp_max que indica o limite máximo para o índice final de p;

– Variável passo referente ao passo da tabela.

Note-se que, neste programa, vai-se considerar os índices finais dos vectores p e q e não o número de elementos desses vectores.

Note-se também que fp e fp_max não têm o mesmo significado. Como já foi referido, conjugar duas tabelas no vector p pode dar origem a valores de fp que estão demasiado além da capacidade instalada (em P.U.) inerente às duas tabelas, o que não faz sentido. Sendo assim, optou-se por considerar esses casos como concentrados em fp_max, ou seja, numa capacidade fora de serviço (em P.U.) inteira, imediatamente a seguir à soma da capacidade instalada (em P.U.) das duas tabelas.

### 4.2.4.3 – Fluxogramas do programa 2

Na fig. 4.12, apresenta-se o fluxograma do módulo principal. O programa dá a possibilidade de se poder calcular o LOLE mais do que uma vez com passos de tabela diferentes. A repetição dos cálculos permite comparar a influência do passo no resultado. Se o passo for igual ao máximo divisor comum das potências dos geradores, a tabela possui todos os estados possíveis do sistema, o que faz com que não haja erro de arredondamento de tabela. No entanto, este passo pode ser demasiado pequeno e pode originar vectores demasiado extensos para a memória disponível.

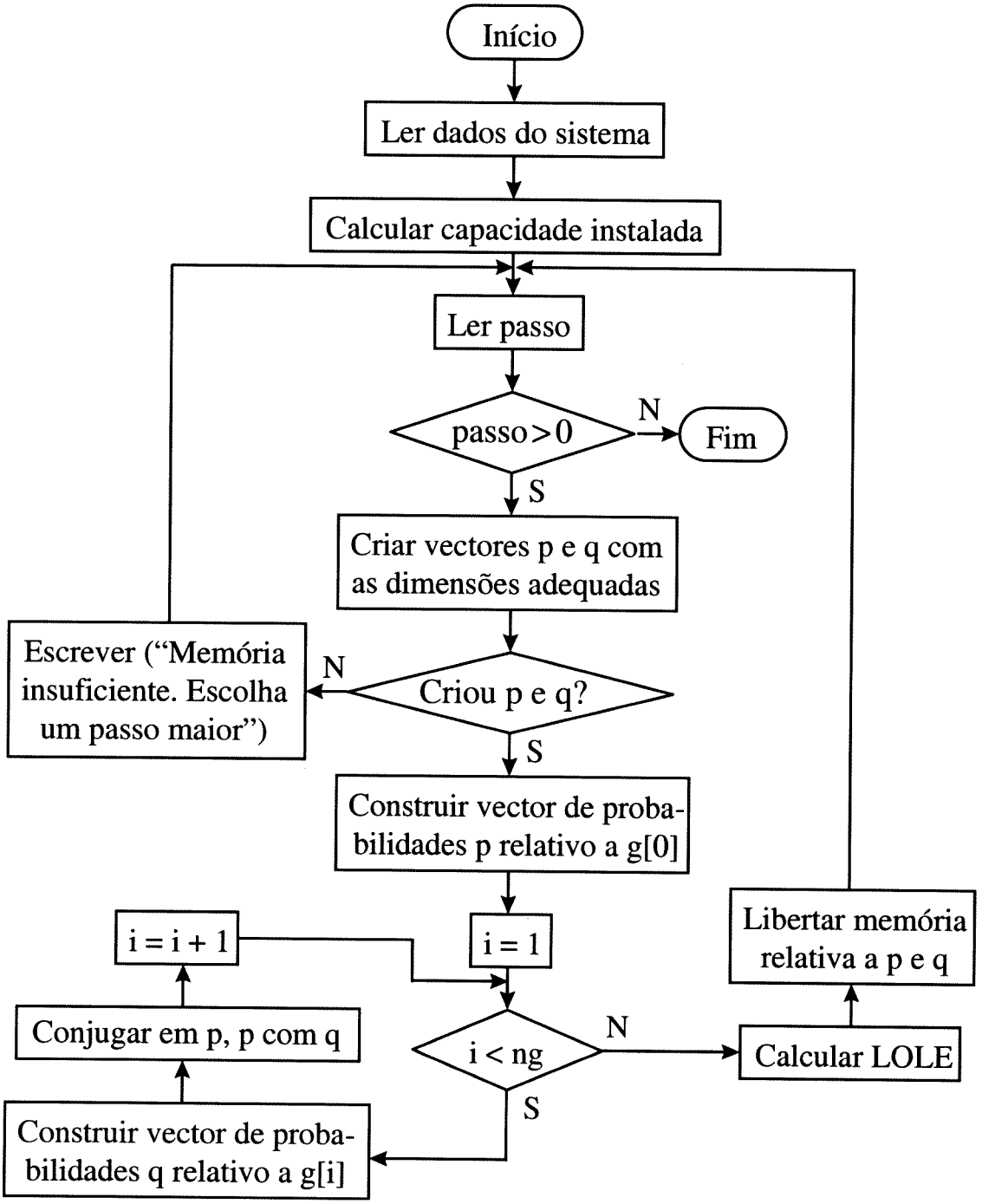



**fig. 4.12 – Módulo principal do programa 2**

Os módulos para ler dados do sistema e calcular a capacidade instalada são iguais aos dos fluxogramas das figs. 4.4 e 4.5, respectivamente.

Na fig. 4.13, encontra-se o fluxograma do módulo que constrói a tabela normalizada relativamente a um conjunto de geradores do mesmo tipo. Para compreender este módulo, tem-se que admitir que os valores de pot, r e n são dados previamente e representam respectivamente a potência de cada gerador em

P.U., FOR de cada gerador e número de geradores do conjunto. O vector v é um vector de probabilidades que tanto se pode identificar com o vector p como com o vector q, dependendo da chamada a este módulo.

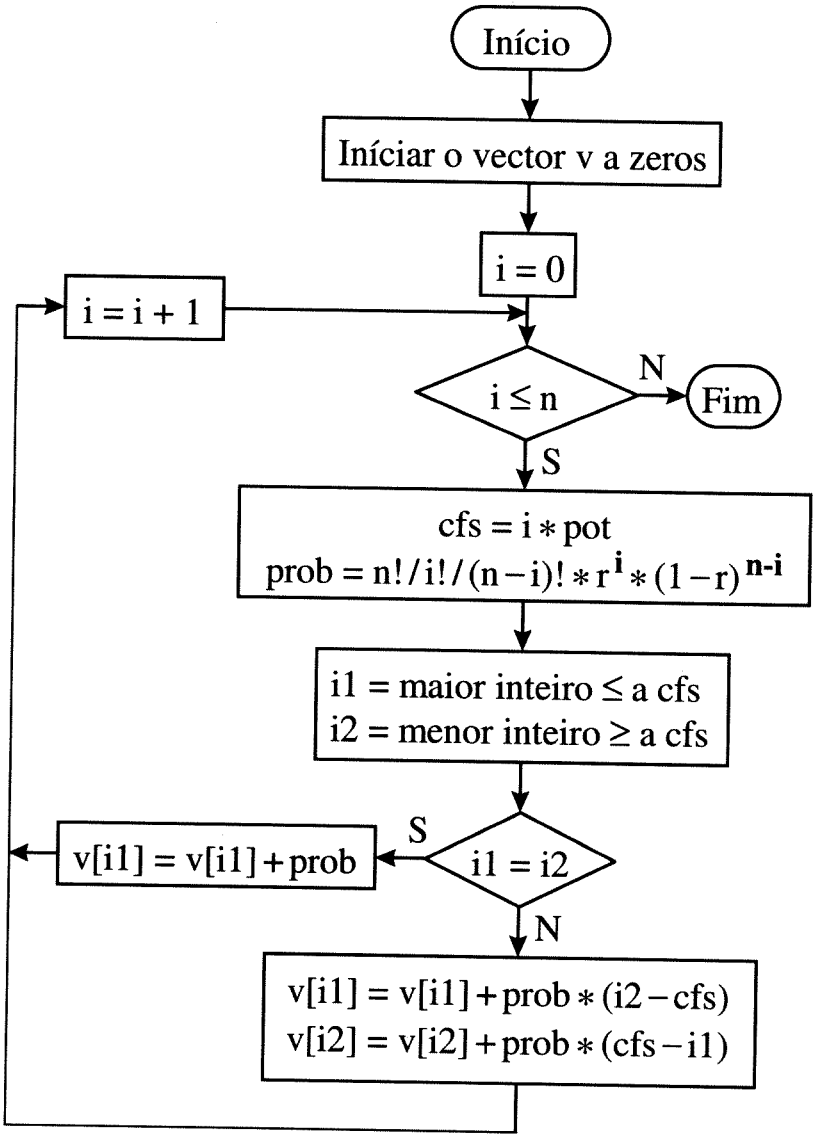



**fig. 4.13 – Módulo de construção da tabela normalizada de um conjunto de geradores do mesmo tipo**

Na fig. 4.14, encontra-se o fluxograma do módulo que conjuga, no vector p, os vectores p e q. Os vectores p e q, bem como as variáveis fp, fq e fp_max são dados previamente e o respectivo significado já é conhecido.

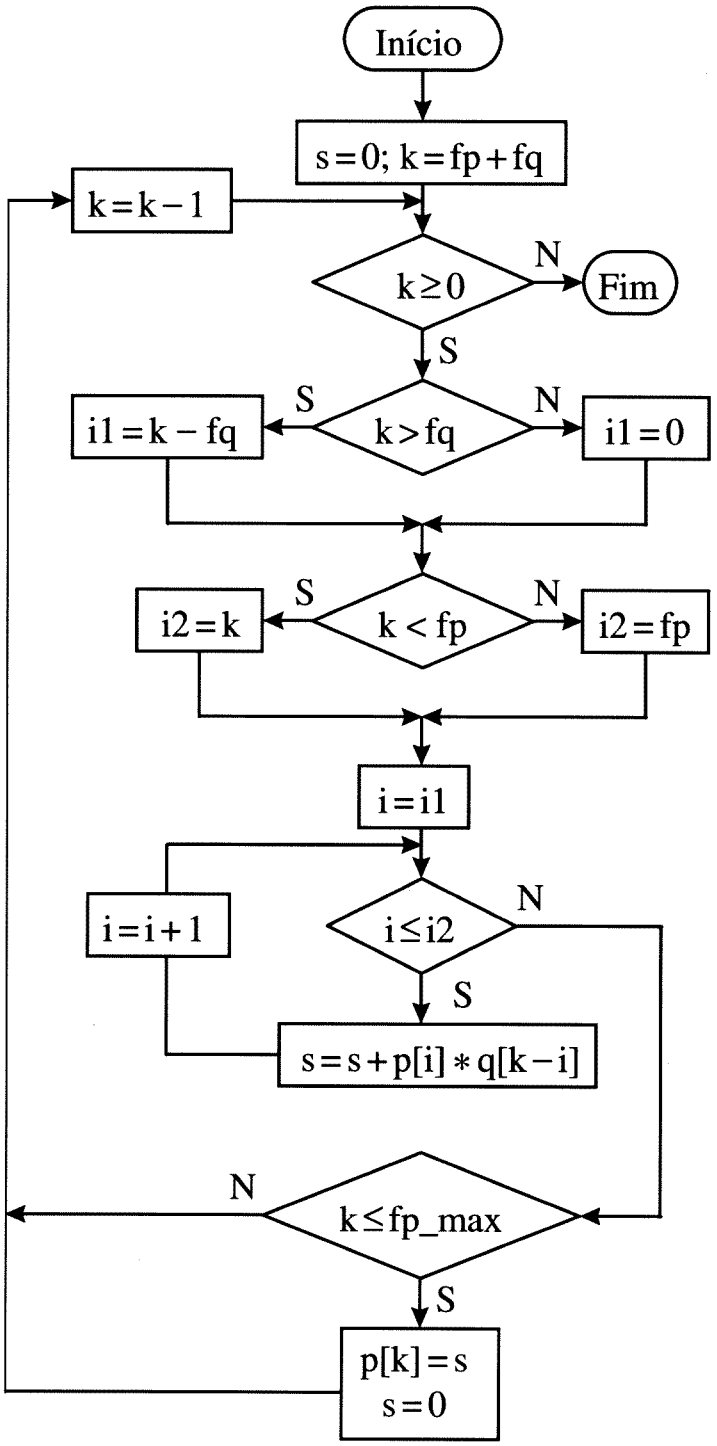



**fig. 4.14 – Módulo de conjugação de tabelas normalizadas**

O módulo que calcula o LOLE é idêntico ao da fig. 4.9, embora haja pequenas diferenças originadas pelo facto das capacidades fora de serviço estarem associadas ao índice dum vector e serem em P.U. (base igual ao passo da tabela). Para calcular o LOLE com as potências em unidades do Sistema Internacional, admita-se que o passo foi fornecido previamente, bem como o vector p e a respectiva variável fp, o mínimo e o

máximo (em P.U.) do diagrama de cargas e a capacidade instalada. Na fig. 4.15, pode-se ver o respectivo fluxograma.

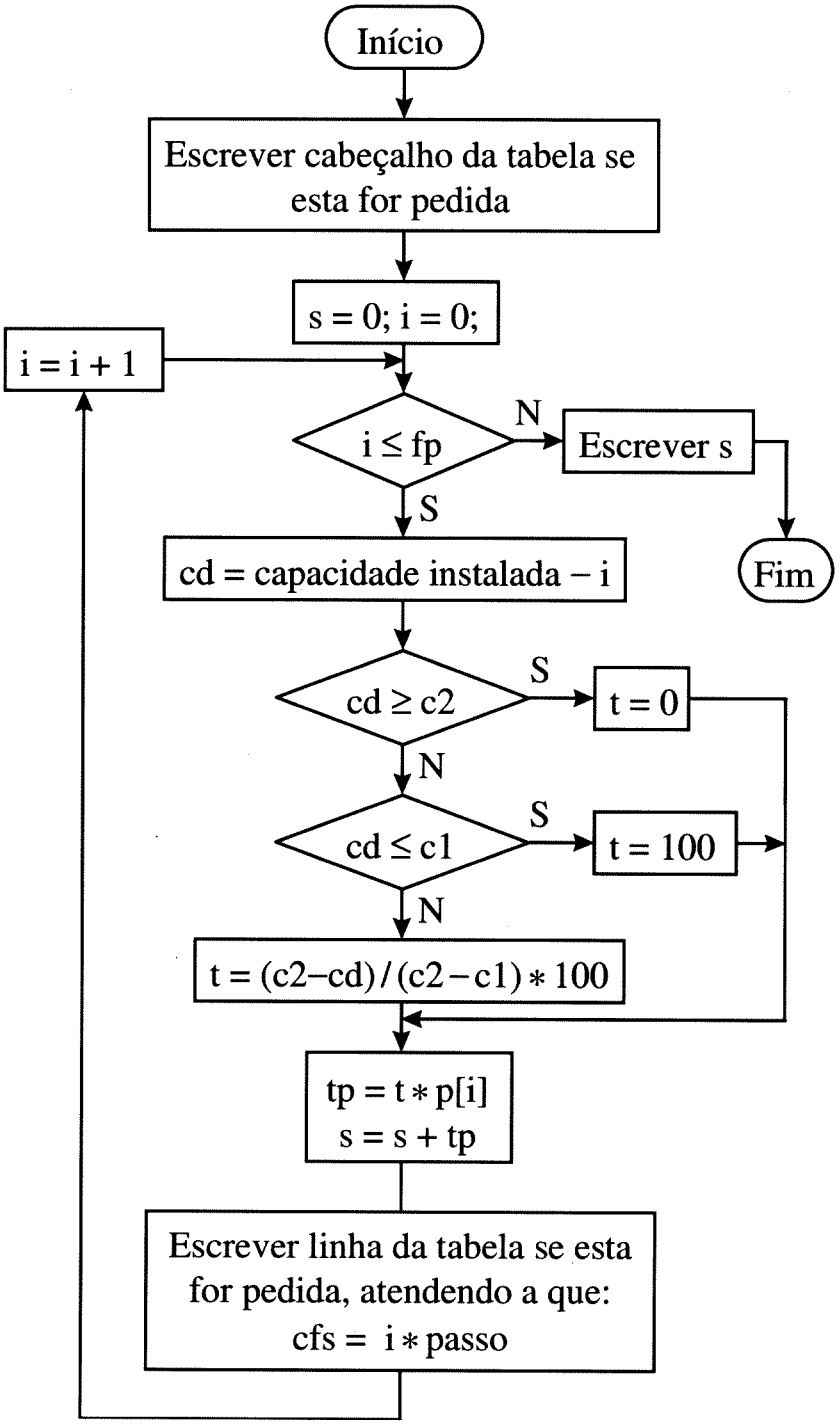



**fig. 4.15 – Módulo do cálculo do LOLE com tabela normalizada das probabilidades das capacidades fora de serviço**

## 4.2.4.4 – Resultados dos testes do programa 2

Na tabela 4.4, apresenta-se os resultados obtidos pelo programa 2 para os três sistemas de teste considerados anteriormente (parágrafo 4.2.1).

| | Passo | LOLE (%) |
|---|---|---|
| Sistema 1 | 50,000 | 0,436502 |
| Sistema 2 | 5,000 | 1,663596 |
| Sistema 2 | 11,000 | 1,660171 |
| Sistema IEEE | 40,000 | 0,517176 |
| Sistema IEEE | 60,000 | 0,529343 |
| Sistema IEEE | 100,000 | 0,533633 |
| Sistema IEEE | 68,100 | 0,523228 |
| Sistema IEEE | 34,050 | 0,517653 |
| Sistema IEEE | 17,025 | 0,515985 |
| Sistema IEEE | 1,000 | 0,515426 |

**Tabela 4.4 – Resultados obtidos pelo programa 2**

Repare-se que os dois primeiros resultados são iguais aos obtidos através do programa 1. Isto acontece porque todas as potências dos geradores são múltiplas do passo escolhido e, assim sendo, todos os estados do sistema estão contemplados na tabela, o que evita os arredondamentos de tabela e respectivos erros. Como a escolha do passo 11MW para o sistema 2 já dá origem a erros de arredondamento da tabela, obtém-se um valor ligeiramente diferente para o LOLE. Para o sistema IEEE, conseguiu-se obter um valor teórico exacto quando foi utilizado o passo de 1MW, pois a tabela vai contemplar todos os estados, pela mesma razão já focada anteriormente. Repare-se que, em geral, passos maiores dão origem a erros maiores. Note-se também que é conveniente que o passo seja um submúltiplo da capacidade instalada.

Obteve-se ainda a tabela para o sistema 1 com passo de 50MW que é igual à obtida através do programa 1. A tabela obtida para sistema 2 com passo de 5MW, continua a ser equivalente à obtida através do programa 1. Na tabela 4.5, está a tabela do sistema 2 com passo de 11MW.

| CFS | Prob | T (%) | T * Prob (%) |
|---|---|---|---|
| 0,00 | 0,933394 | 0,00 | 0,000000 |
| 11,00 | 0,034499 | 0,00 | 0,000000 |
| 22,00 | 0,031011 | 50,00 | 1,550536 |
| 33,00 | 0,000909 | 100,00 | 0,090892 |
| 44,00 | 0,000184 | 100,00 | 0,018418 |
| 55,00 | 0,000003 | 100,00 | 0,000326 |
| | 1,000000 | | 1,660172 |

**Tabela 4.5 – Tabela arredondada do sistema 2 com passo de 11MW**

## 4.2.5 – Programa 3

### 4.2.5.1 – Método de Simulação de Monte Carlo

No programa 3, vai-se usar o Método de Simulação de Monte Carlo, em que o sorteio consiste numa amostragem aleatória de estados do sistema (sorteio não cronológico) [6]. Para reduzir a variância e acelerar a convergência, vai-se usar variáveis antitéticas (ver parágrafo 3.6.3). O cálculo do LOLE vai ser acompanhado duma visualização gráfica do mesmo, o que permite ao utilizador analisar se é atingida estabilidade no valor estimado .

O LOLE é dado por:

$$LOLE = \sum_{i=1}^{m} t_i \cdot P_i \qquad (4.10)$$

em que:

– $t_i \rightarrow$ Tempo durante o qual a carga não é alimentada se o sistema estiver no i-ésimo estado;

– $P_i \rightarrow$ Probabilidade de ocorrer o i-ésimo estado;

– $m \rightarrow$ Número de estados do sistema.

Note-se que na fórmula 4.1 considera-se capacidades fora de serviço e na fórmula 4.10 considera-se estados do sistema,

mas ambas as fórmulas são válidas, porque a interrupção provocada por uma capacidade fora de serviço é o resultado de todas as interrupções dos estados do sistema que dão origem a essa capacidade fora de serviço.

Quando fazemos a simulação pelo Método de Monte Carlo, considera-se uma amostra de N estados sorteados, admitindo ser representativa dos m estados do sistema. Nesta simulação, a probabilidade de ocorrer o i-ésimo estado do sistema é dada por:

$$P_i = \frac{n_i}{N} \qquad \text{sendo} \qquad \sum_{i=1}^{m} n_i = N$$

em que $n_i$ é o número de vezes que ocorreu o i-ésimo estado do sistema.

Sendo assim, o cálculo do LOLE é então dado por:

$$LOLE = \sum_{i=1}^{m} t_i \cdot P_i = \sum_{i=1}^{m} t_i \frac{n_i}{N} = \frac{1}{N} \sum_{i=1}^{m} t_i \cdot n_i \qquad (4.11)$$

Por outro lado, note-se também que:

$$\sum_{i=1}^{m} t_i \cdot n_i = \sum_{j=1}^{N} t_j'$$

em que $t_j'$ é o tempo durante o qual a carga não é alimentada se o sistema estiver no estado correspondente ao j-ésimo sorteio (não corresponde necessariamente o j-ésimo estado). Assim sendo, a fórmula 4.11 toma aspecto seguinte:

$$LOLE = \frac{\sum_{j=1}^{N} t_j}{N}, \qquad (4.12)$$

Esta é a fórmula usada para calcular o LOLE.

### 4.2.5.2 – Estrutura de dados do programa 3

As principais estruturas de dados do programa 3 são as seguintes:

– Vector g igual ao que foi definido no parágrafo 4.2.3.2;

– Variável ng que representa o número de elementos do vector g;

– Variável n que representa a dimensão da amostra;

– Variáveis c1 e c2 onde se armazena respectivamente o mínimo e o máximo do diagrama de cargas classificado;

– Variável max que indica o valor máximo do LOLE que pode ser representado graficamente (ordenada máxima).

O programa desenha gráficos em que as abcissas representam o tamanho da amostra até ao momento e as ordenadas representam o valor do LOLE. É dada ao utilizador a possibilidade de alterar a escala das ordenadas, através da leitura de um novo valor de ordenada máxima, além da possibilidade de alterar o tamanho da amostra. Desta forma, através da observação de gráficos, o utilizador consegue descobrir uma dimensão da amostra para a qual há convergência do valor estimado para o LOLE.

### 4.2.5.3 – Fluxogramas do programa 3

A fig. 4.16, representa o fluxograma do módulo principal deste programa.

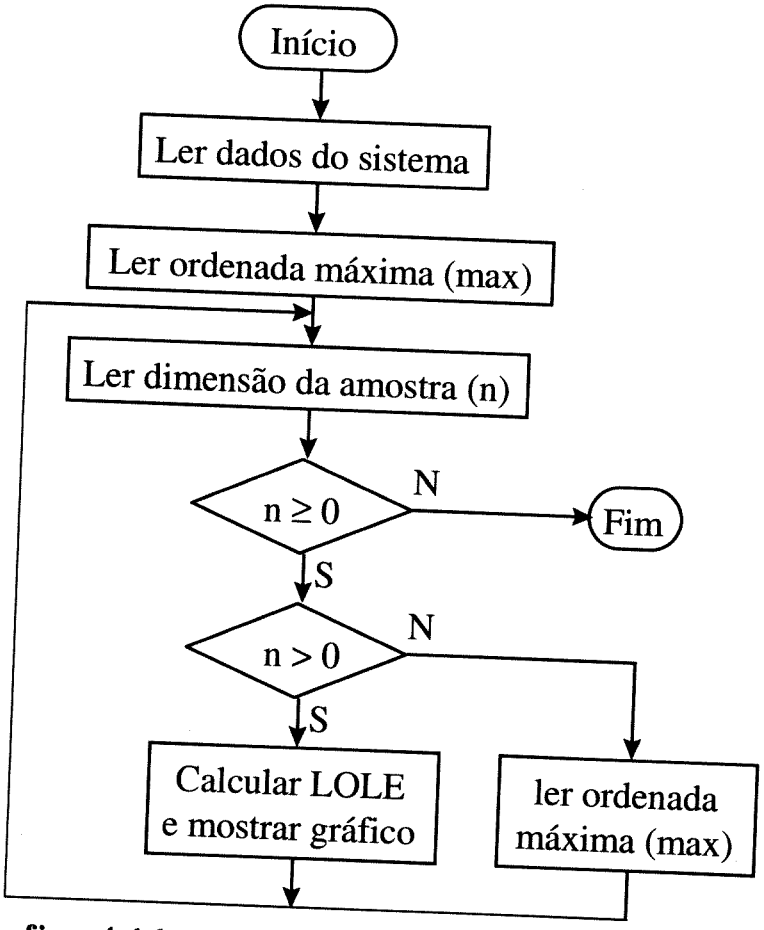



**fig. 4.16 – Módulo principal do programa 3**

O módulo ler dados do sistema já foi desenvolvido na fig. 4.4.

Nas figs. 4.17 e 4.18 está o módulo que calcula o LOLE e mostra o respectivo gráfico. O vector g e as variáveis ng, c1 e c2 já vêm definidas do módulo principal.

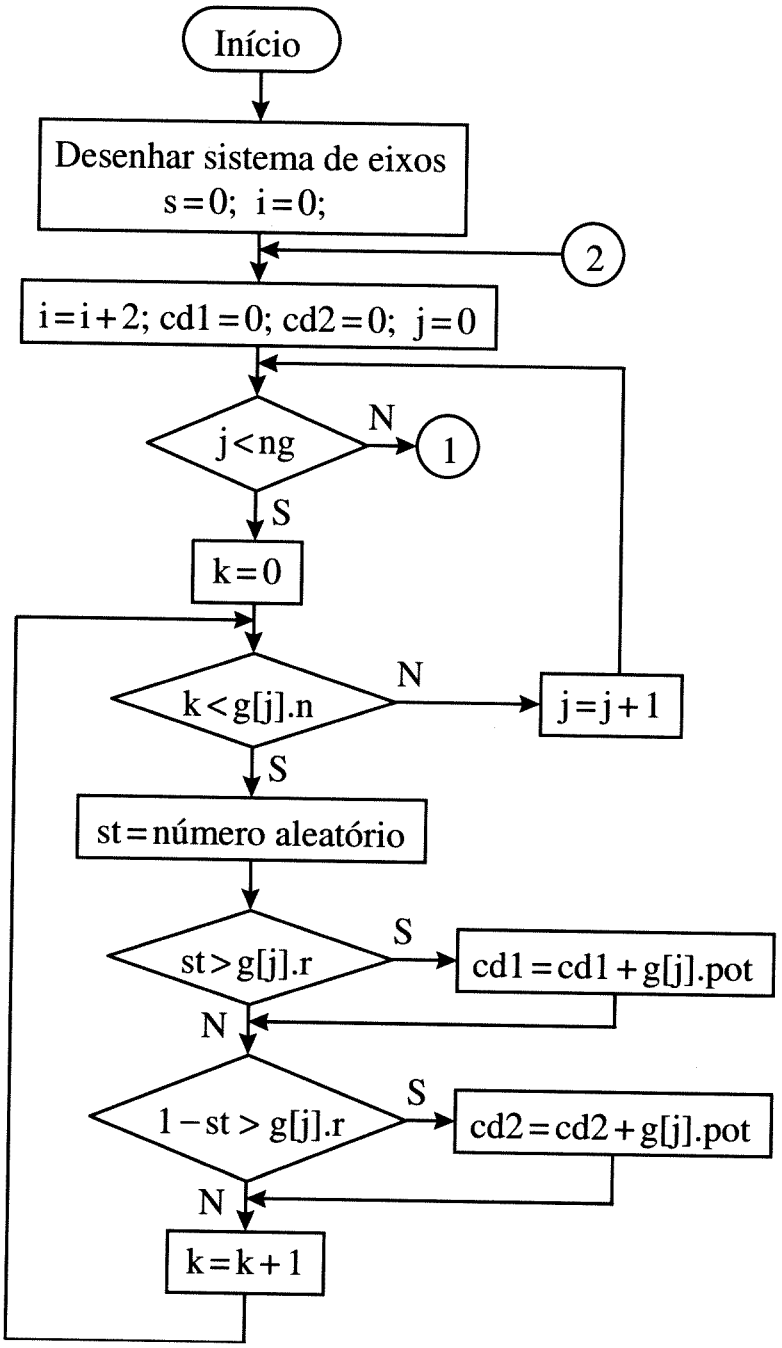



**fig. 4.17 – Módulo do calculo do LOLE e respectiva visualização gráfica**

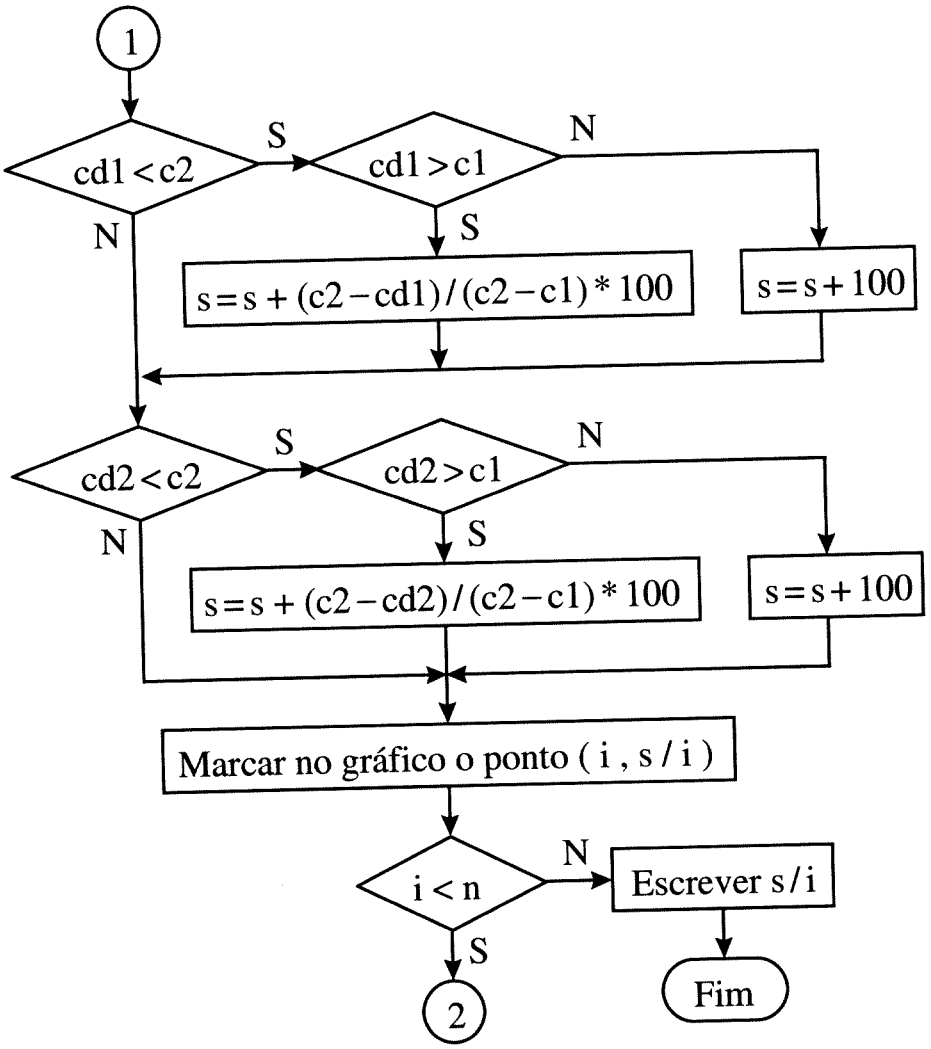



**fig. 4.18 – Conclusão do módulo da fig. 4.17**

## 4.2.5.4 – Resultados dos testes do programa 3

Na tabela 4.6, apresenta-se os resultados obtidos pelo programa 3 para os três sistemas de teste considerados anteriormente (parágrafo 4.2.1).

| | Dimensão da amostra | LOLE (%) | Valor analitíco correcto (%) |
|---|---|---|---|
| **Sistema 1** | 1000000 | 0,437 | 0,436502 |
| **Sistema 2** | 1000000 | 1,665 | 1,663596 |
| **Sistema IEEE** | 1000000 | 0,515 | 0,515426 |

**Tabela 4.6 – Resultados obtidos pelo programa 3**

Para todos os sistemas de teste, o programa 3 sorteou um milhão de estados (incluindo os estados sorteados de modo

antitético) e obteve um valor do LOLE que é uma boa aproximação do valor analítico correcto. Através dos gráficos das figs. 4.19, 4.20 e 4.21, pode-se ver que a estabilidade do cálculo foi atingida em todos os casos, o que prova que o tamanho escolhido para as amostras é suficiente.

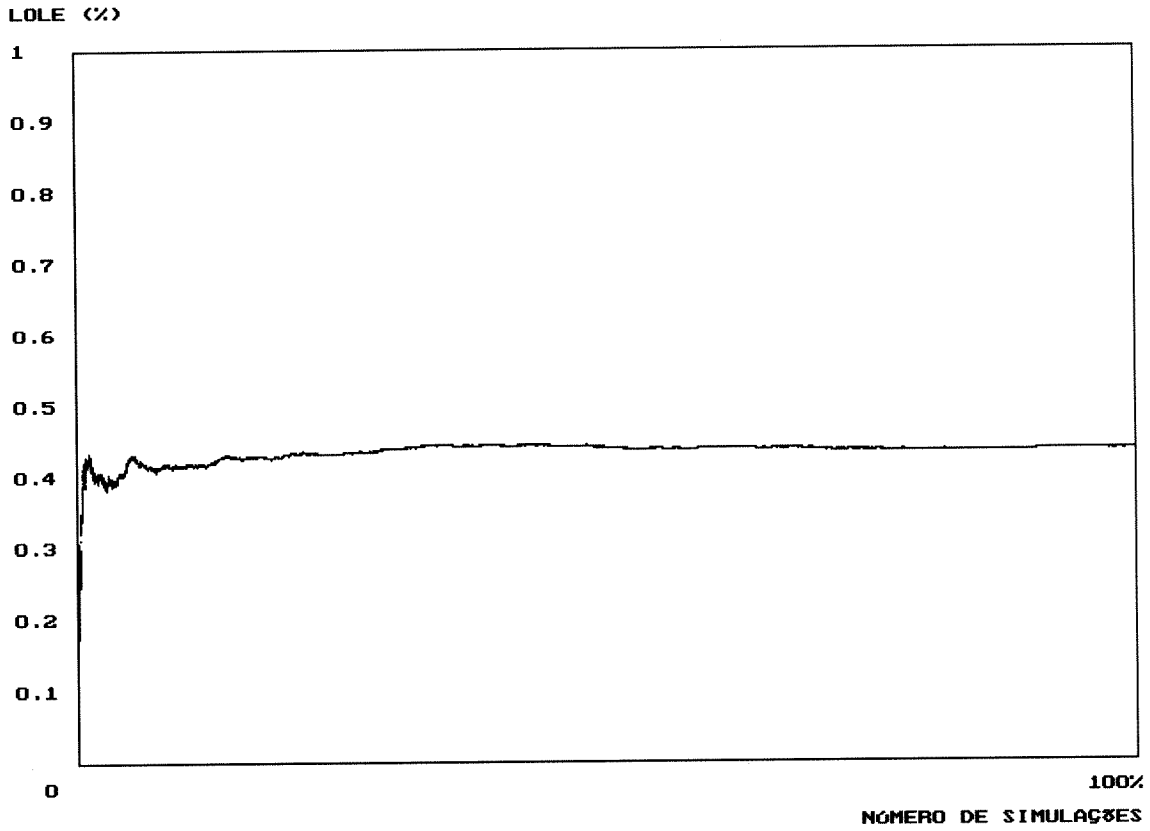



**fig. 4.19 – Visualização gráfica do cálculo do LOLE do sistema 1**

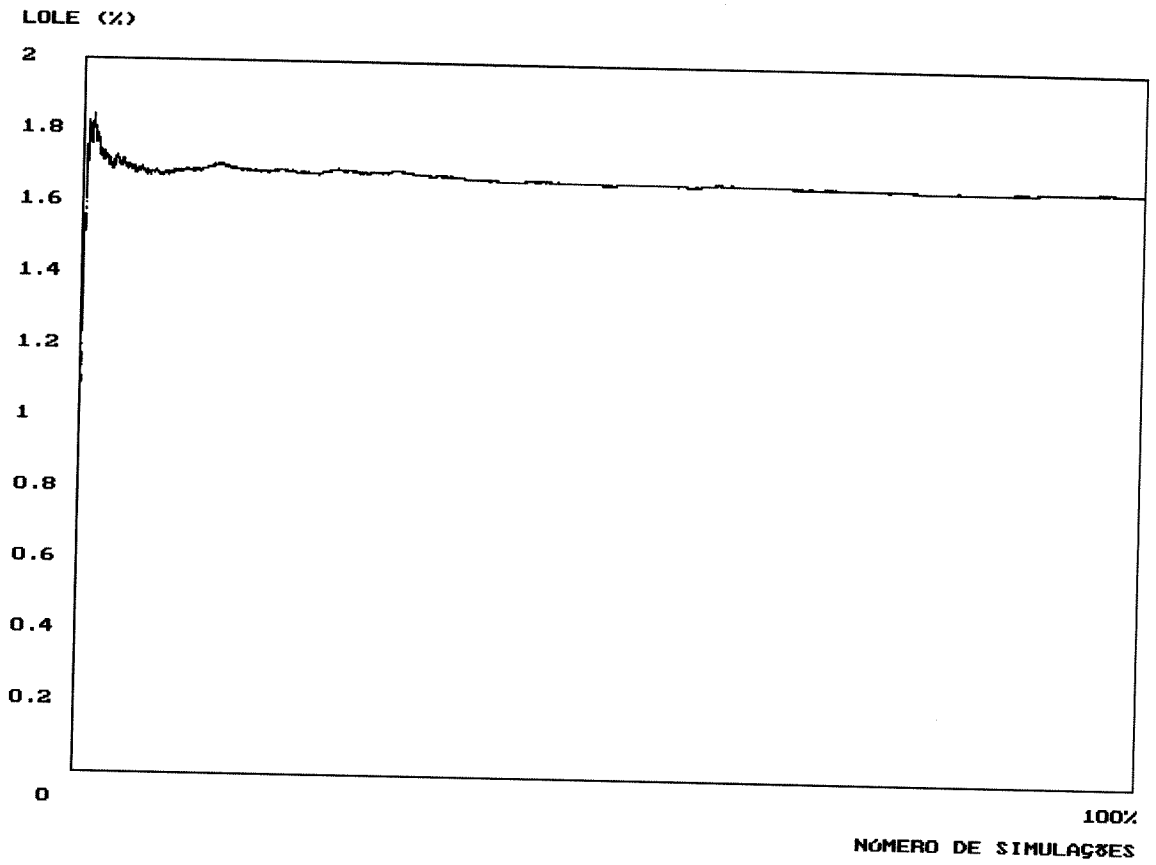



**fig. 4.20 – Visualização gráfica do cálculo do LOLE do sistema 2**

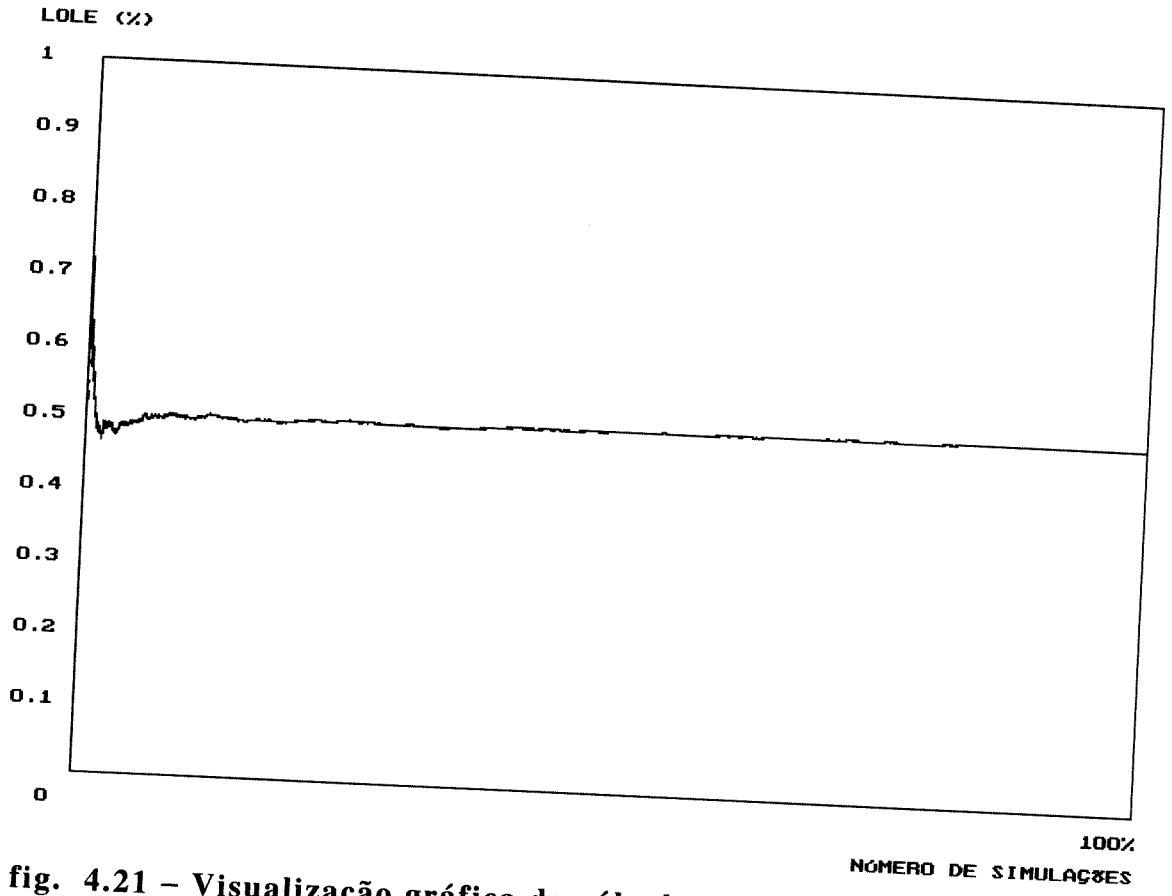



**fig. 4.21 – Visualização gráfica do cálculo do LOLE do sistema IEEE**

## 4.2.6 – Programa 4

### 4.2.6.1 – Método usado no programa 4

O cálculo do LOLE é feito da mesma maneira que no programa 3 (ver parágrafo 4.2.5.1), mas o critério de convergência deixa de ser estabelecido pelo utilizador através da visualização de um gráfico e passa a ser automático [6;24].

O critério utilizado baseia-se no que foi descrito no parágrafo 3.3. O objectivo é obter uma amostra em que o coeficiente de variação ($\beta$) da grandeza a estimar seja inferior a um dado valor (ver expressão 3.5). De acordo com a expressão 3.5, é necessário calcular a média e a variância da grandeza a estimar. Para calcular a variância sempre que é feito um novo sorteio, é necessário recalcular a soma de todos os desvios quadráticos, pois a média pode ter sido alterada com este sorteio. Tal método pode ser pesado em termos computacionais.

Mas, por outro lado, já se viu através dos gráficos obtidos pelo programa anterior que o cálculo do LOLE acaba por estabilizar. Portanto, pode-se pensar que à medida que vão

sendo realizados sorteios, a média acaba por ser um valor praticamente fixo e a variância diminui. Sendo assim, vai ser considerado que a convergência é alcançada se o coeficiente de variação dos últimos sorteios obedecer ao estipulado. No entanto, a média em relação à qual a variância é calculada continua a ser a média global (ou aproximação desta) e não a média dos últimos sorteios. Portanto, tem-se os passos seguintes:

– Guardar o valor da média actual;

– Realizar mais um determinado número de sorteios e calcular a soma dos desvios quadráticos em relação à média que foi guardada no passo anterior;

– Calcular a variância e o coeficiente de variação em relação a estes últimos sorteios, tendo em conta os valores obtidos nos passos anteriores;

– Verificar se o coeficiente de variação alcançou o estipulado.

A grande vantagem é que este método, em termos computacionais, é bastante menos pesado.

### 4.2.6.2 – Estrutura de dados do programa 4

As principais estruturas de dados do programa 4 são as seguintes:

– Vector g e variáveis ng, c1 e c2 com o significado já conhecido;

– Variável n que representa a dimensão máxima da amostra;

– Variável cv que representa o coeficiente de variação;

– Variável d que representa a duração (número de simulações consecutivas) do teste de estabilidade;

– Variável op que representa a opção do utilizador.

### 4.2.6.3 – Fluxogramas do programa 4

A fig. 4.22, representa o fluxograma do módulo principal.

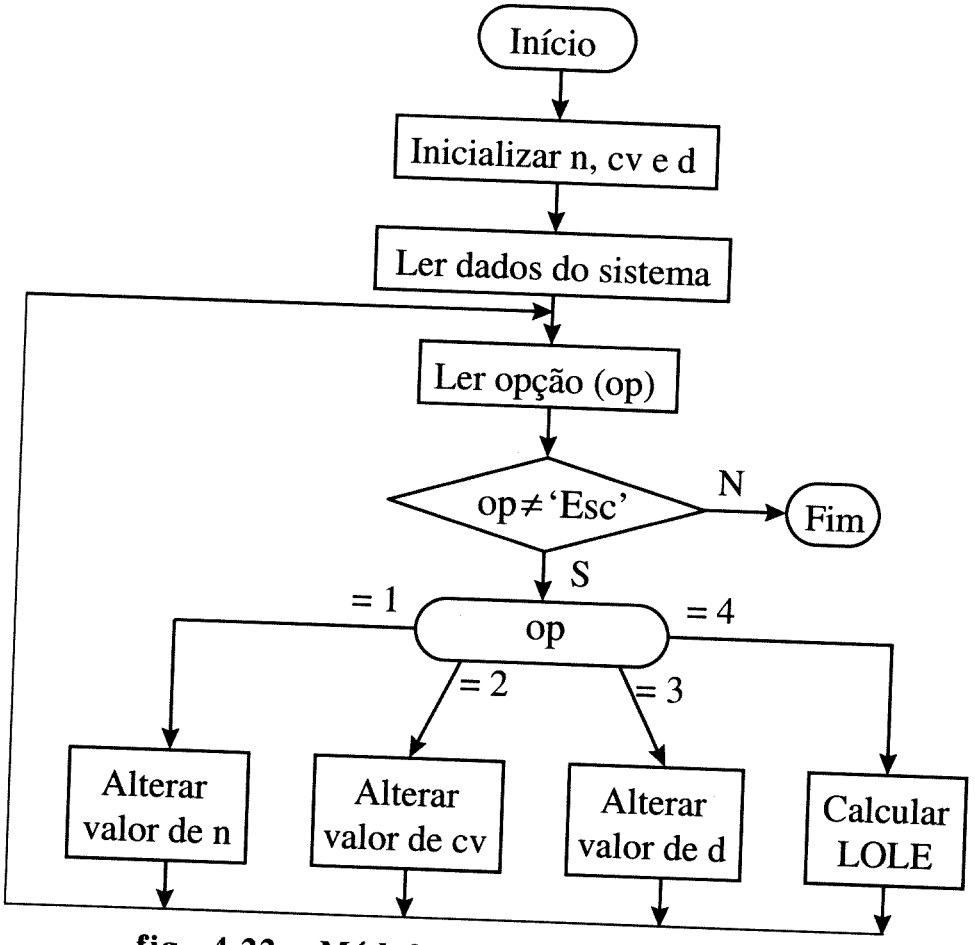



**fig. 4.22 – Módulo principal do programa 4**

Este programa é bastante versátil, porque permite alterar a dimensão máxima da amostra, o coeficiente de variação ou a duração do teste de estabilidade antes de se mandar calcular o LOLE. Desta forma, consegue-se, através duma escolha adequada desses parâmetros, obter resultados suficientemente precisos.

O módulo que lê os dados do sistema já é conhecido. Na fig. 4.23 pode-se ver o fluxograma do módulo que calcula o LOLE.

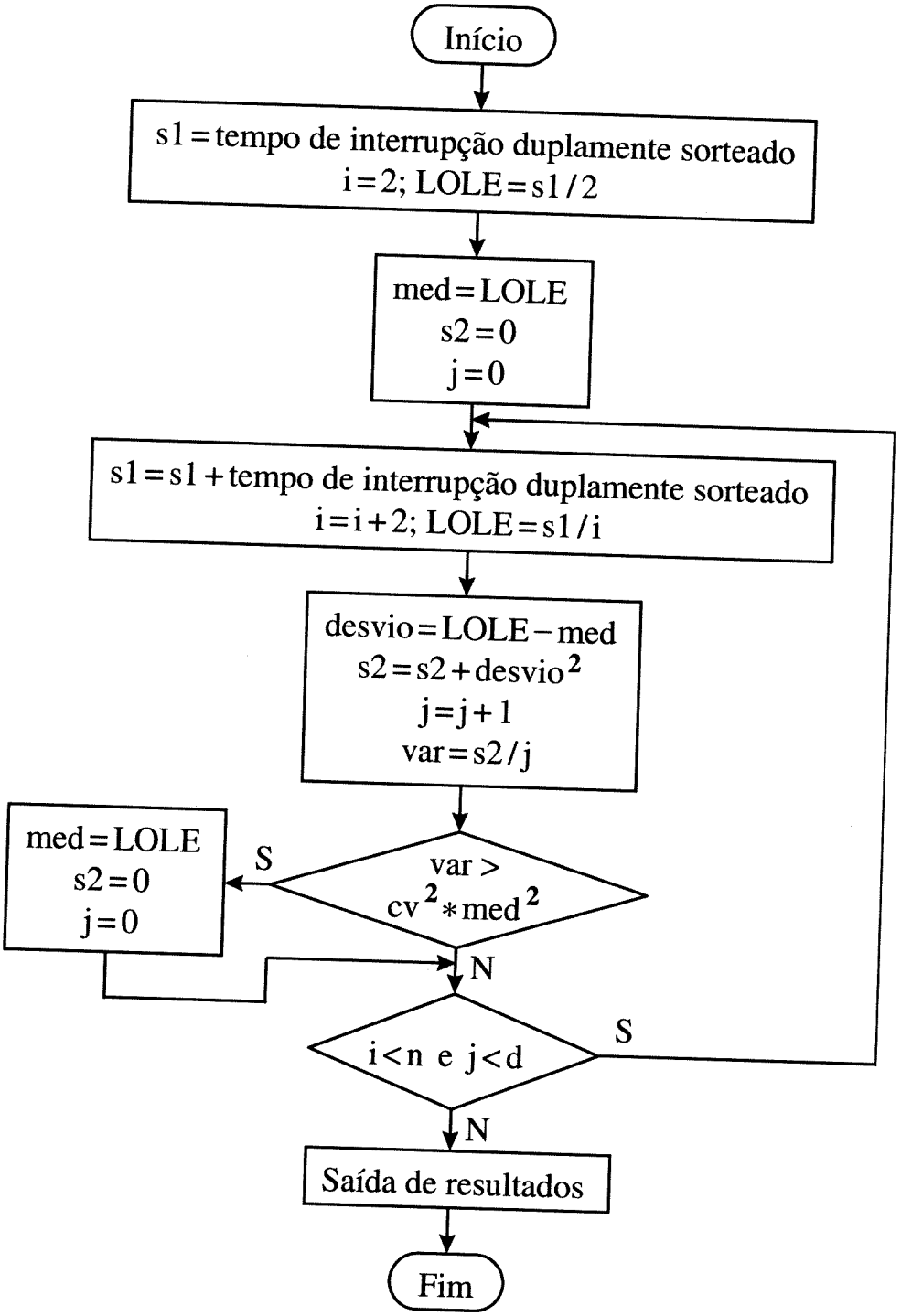



**fig. 4.23 – Módulo do cálculo do LOLE**

Nas figs. 4.24 e 4.25, apresenta-se o módulo que faz o duplo sorteio de tempo de interrupção. Este módulo sorteia um estado do sistema e calcula o tempo de interrupção que esse estado causa na alimentação das cargas, voltando a repetir o mesmo cálculo para o estado antitético. Por último, soma os dois tempos de interrupções obtidos e devolve esta soma ao módulo calcular LOLE.

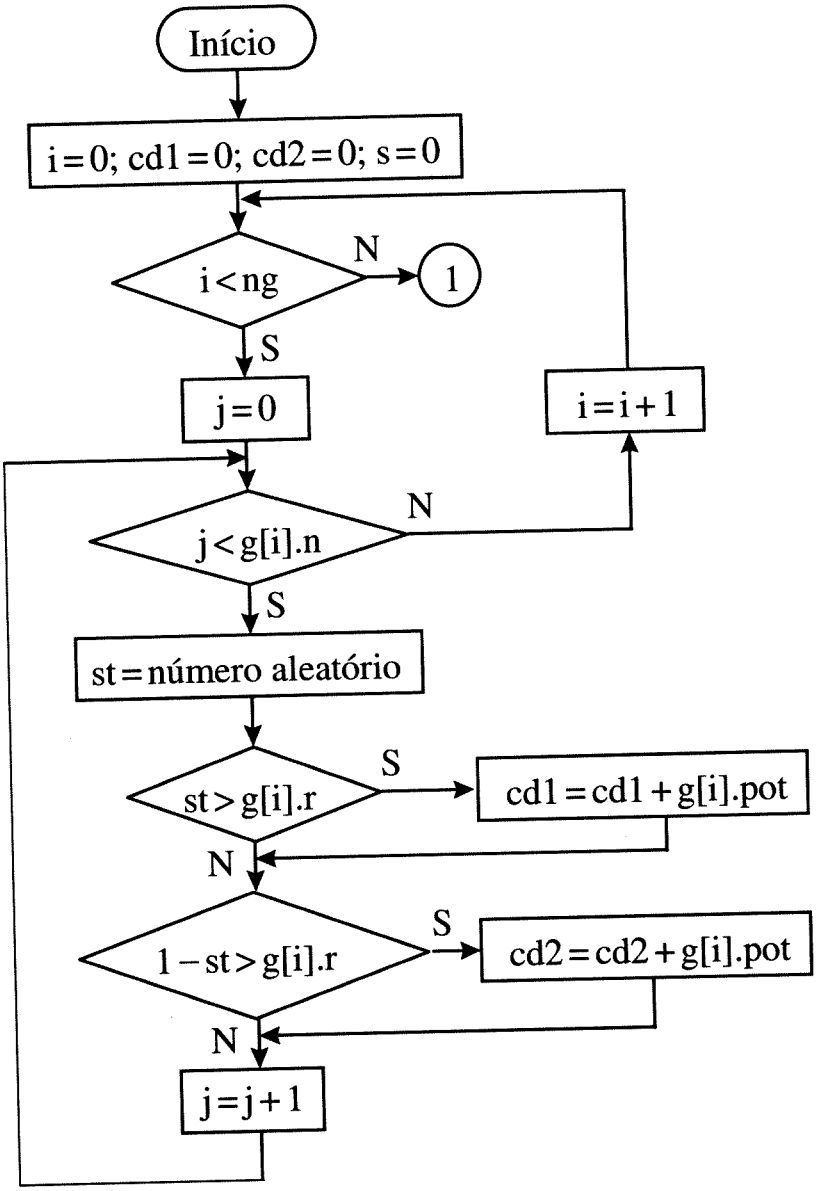



**fig. 4.24 – Módulo do cálculo da soma dos tempos de interrupção causados por um estado sorteado e pelo seu estado antitético.**

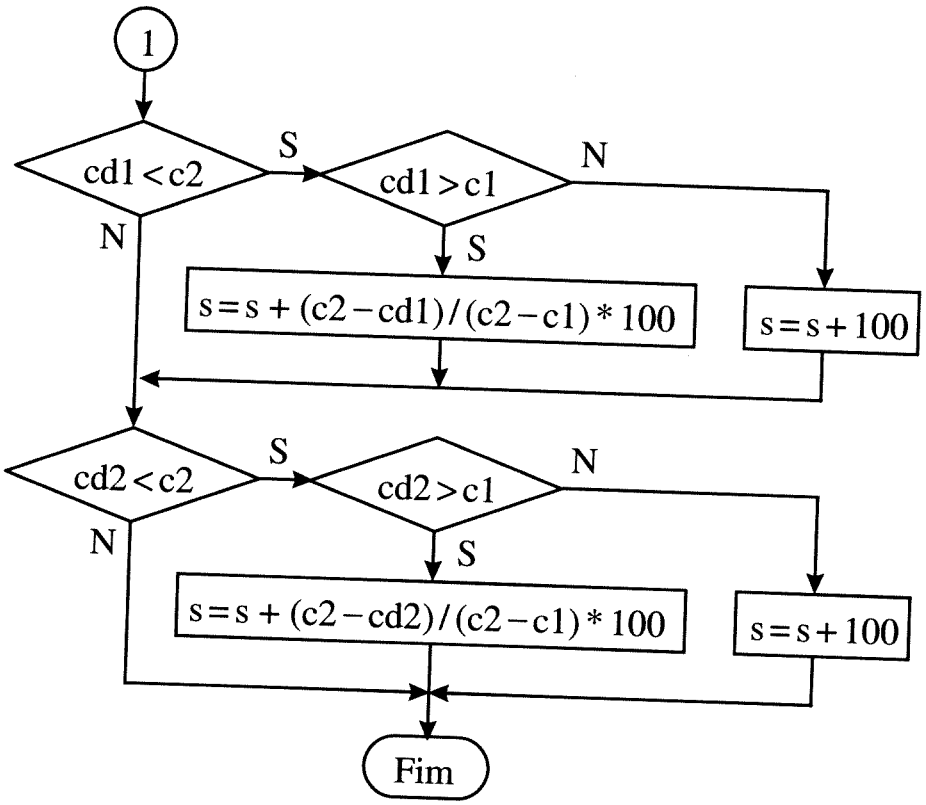



**fig. 4.25 – Conclusão do módulo da fig. 4.24**

## 4.2.6.4 – Resultados dos testes do programa 4

Na tabela 4.7, apresenta-se os resultados obtidos pelo programa 4 para os três sistemas de teste considerados anteriormente (parágrafo 4.2.1).

| | Sistema 1 | Sistema 2 | Sistema IEEE |
|---|---|---|---|
| Coeficiente de variação | 0,001 | 0,001 | 0,001 |
| Duração da estabilidade | 20000 | 20000 | 20000 |
| Dimensão da amostra | 704068 | 354876 | 562104 |
| Dimensão máxima da amostra | 1000000 | 1000000 | 1000000 |
| LOLE (%) | 0,436 | 1,660 | 0,512 |
| Valor analitíco correcto (%) | 0,436502 | 1,663596 | 0,515426 |

**Tabela 4.7 – Resultados obtidos pelo programa 4**

Pode-se ver na tabela 4.7 que os valores do LOLE obtidos pelo programa 4 continuam a ser uma boa aproximação dos valores correctos. Em todos os casos, a dimensão da amostra foi sempre inferior à sua dimensão máxima, o que significa que o programa 4 terminou porque foi verificado o critério de

estabilidade. Caso tal facto não se tivesse verificado, poderia repetir-se os cálculos com dimensões máximas maiores.

## 4.3 – Conclusões

O Método de Monte Carlo consegue obter uma precisão suficiente para o fim em vista. Por vezes, é tanto ou mais preciso que os métodos analíticos se nestes forem introduzidas aproximações. A título de exemplo, pode-se ver que o programa 1 calculou do LOLE do sistema IEEE pelo método de truncamento da tabela das probabilidades de ocorrência das capacidades fora de serviço e obteve um valor menos preciso do que aqueles que foram obtidos pelos programas 3 e 4, que usam o Método de Monte Carlo.

O Método de Monte Carlo conduz a programas bastante mais simples. Note-se que foi necessário um certo esforço de optimização, quer em termos de aproveitamento de memória quer em termos de código, para que os programas que calculam o LOLE por métodos analíticos recorressem o mínimo possível a aproximações.

Um inconveniente do Método de Monte Carlo pode ser o tempo computacional, mas como estes cálculos não são para serem efectuados em tempo real, este problema não se põe.

# 5. FIABILIDADE DO SISTEMA COMPOSTO

## 5.1 – Introdução

Neste capítulo vai-se analisar o programa desenvolvido para calcular a Fiabilidade do Sistema Composto usando o Método de Simulação de Monte Carlo. Usa-se um sorteio cronológico através do qual serão sorteadas as transições de estados do sistema [6], tal como foi descrito no parágrafo 3.7.3. Desta forma, será possível a obtenção de gráficos que caracterizam os índices de fiabilidade que se pretende estudar. Os índices de fiabilidade a obter em cada barramento são os seguintes:

– Taxa de avarias;

– Indisponibilidade anual;

– Tempo médio de reparação.

Os índices de fiabilidade serão calculados em cada barramento do sistema, tendo em conta que os barramentos são deslastrados por ordem crescente de prioridade. Durante um estado do sistema, vai-se admitir que a carga em cada barramento se mantém constante e igual ao pico diário no instante em que se iniciou o respectivo estado. Em cada barramento com geração, vai-se considerar que a potência gerada varia continuamente entre um mínimo que é igual à menor potência mínima dos geradores não avariados ligados a esse barramento e um máximo que é igual à soma das potências máximas desses mesmos geradores.

Se a potência pedida pelas cargas for inferior à soma dos mínimos dos barramentos, considere-se que é sempre possível a satisfação das cargas sem violações nos ramos.

Se a potência pedida pelas cargas for superior à potência disponível (soma dos máximos dos barramentos), irão ser efectuados deslastres até que tal situação deixe de se verificar. Nestas situações em que há pelo menos um deslastre, considera-se que as restantes cargas serão satisfeitas sem haver violações dos fluxos dos ramos.

Finalmente, se a soma das cargas se situar, à partida, entre a soma dos mínimos dos barramentos e a potência disponível, irá ser feito um despacho e a análise dos fluxos nos ramos. Se houver uma violação de fluxo num ramo, vai-se fazer redespachos de modo a tentar corrigir essa mesma violação. Este processo será aplicado a todos os fluxos violados, mas, para que o processo não caia em ciclo vicioso, admita-se que uma violação só terá uma oportunidade de ser corrigida. Se a tentativa de correcção do fluxo num ramo não for bem sucedida ou se esse fluxo já foi corrigido e voltou a ficar violado devido a correcções posteriores efectuadas noutros ramos, então será necessário proceder à realização de um deslastre. A partir deste momento, como já houve um deslastre, considera-se que as restantes cargas são alimentadas sem haver violações nos ramos, como já foi referido anteriormente. O cálculo dos fluxos nos ramos será feito através do modelo DC [23]. A correcção do fluxo num ramo será feita tendo em conta os coeficientes de sensibilidade.

No caso do sistema formar "ilhas", vai ser considerada a avaria total do sistema.

## 5.2 – Cálculo da matriz inversa da matriz das susceptâncias

Sempre que surge um estado do sistema que origine uma reconfiguração da topologia da rede, a matriz das susceptâncias (matriz B) e, consequentemente, a sua inversa são alteradas. Contudo, a inversão da matriz B é bastante penalizante em termos computacionais. Por outro lado, sempre que surge um novo estado em que há uma alteração da topologia da rede, essa alteração reflecte-se apenas num ramo. Assim sendo, é possível, se o sistema se mantiver coeso, obter a matriz inversa de B actual à custa da matriz inversa de B anterior, sem grande esforço computacional [6;20]. Por isso, a matriz B vai ser invertida uma única vez no início e sempre que surgirem "ilhas" no sistema. Como a situação do sistema formar "ilhas" é rara, o peso computacional causado pela inversão da matriz B nessas situações não é significativo.

Quando um ramo avaria ou entra em funcionamento, a matriz B sofre uma alteração que se designa por $\Delta B$. Note-se que a avaria de um ramo de susceptância y é equivalente à inserção em paralelo de um ramo de susceptância -y. Assim

sendo, a avaria ou a entrada em funcionamento de um ramo podem ser ambas tratadas como a inserção de um novo ramo.

Se nenhum dos barramentos i e j a que o ramo está ligado é a referência, então a matriz $\Delta B$ é idêntica à da fig. 5.1, em que y é igual à susceptância do ramo, se o ramo entra em funcionamento, ou igual ao simétrico dessa susceptância, se o ramo avariou.

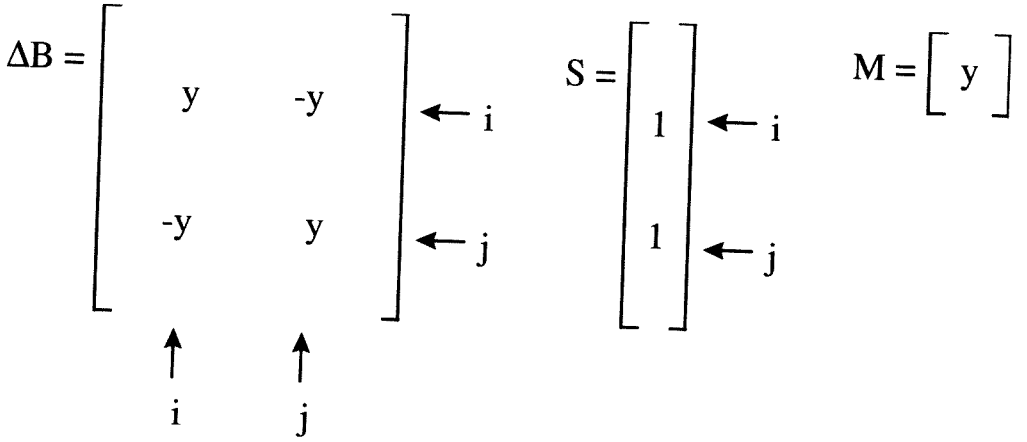



**fig. 5.1 – Matrizes ΔB e M e vector S se nem i nem j são a referência**

Se o ramo está ligado à referência e ao barramento i, então a matriz $\Delta B$ é idêntica à da fig. 5.2.

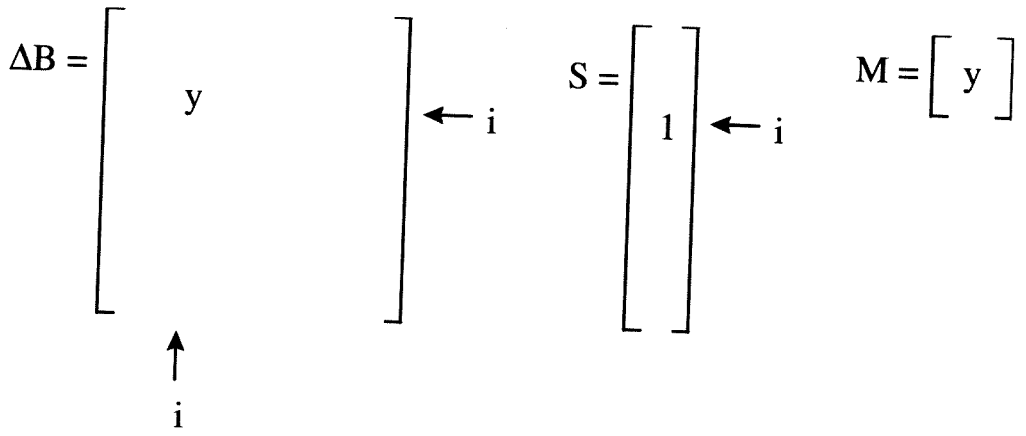



**fig. 5.2 – Matrizes ΔB e M e vector S em que o ramo está ligado à referência e ao barramento i**

Observando as figs. 5.1 e 5.2, pode-se concluir que:

$$\Delta B = S\ M\ S^T \qquad \textbf{(5.1)}$$

Por outro lado, segundo o Lema do Inverso duma Matriz Modificada (LIMM), tem-se que:

$$( B + \Delta B )^{-1} = B^{-1} - B^{-1} ( I + \Delta B \, B^{-1} )^{-1} \Delta B \, B^{-1} \qquad (5.2)$$

em que I é a matriz identidade. Substituindo a expressão 5.1 na expressão 5.2 e desenvolvendo os cálculos [20], obtém-se:

$$( B + \Delta B )^{-1} = B^{-1} - B^{-1} \, S \, C \, S^{T} \, B^{-1} \qquad (5.3)$$

$$C = ( M^{-1} + S^{T} \, B^{-1} \, S )^{-1} \qquad (5.4)$$

Utilizando as expressões 5.3 e 5.4, a matriz $(B+\Delta B)^{-1}$ pode ser obtida sem necessidade de inverter a matriz $B+\Delta B$. Este método é eficiente desde que $\Delta B$ seja esparsa, o que, na realidade, se verifica.

## 5.3 – Despacho

Tal como já foi referido, o programa vai ter necessidade de efectuar um despacho quando a soma das cargas estiver entre a soma dos mínimos dos barramentos e a potência disponível.

Para um dado estado do sistema, o objectivo é encontrar um despacho que permita alimentar o máximo de cargas sem causar violações. Dentro dos despachos que satisfazem o objectivo anterior, iria poder-se pensar em obter o despacho mais económico, mas sob o ponto de vista de fiabilidade o que interessa é encontrar um despacho que minimize os cortes, independentemente de poder existir outro despacho que faça o mesmo e que seja mais económico. Portanto, o despacho de arranque para analisar um estado, não vai basear-se num critério económico, mas sim numa distribuição proporcional da potência pedida por todos os barramentos com produção.

Assim, tem-se:

$$Smax = \Sigma \, max_i$$

$$Smin = \Sigma \, min_i$$

$$Ped = \sum pc_i$$

$$Smin \leq Ped \leq Smax$$

$$pg_i = min_i + \Delta pg_i \qquad (5.5)$$

em que:

– $max_i$ é a potência máxima que é possível gerar no barramento i;

– $min_i$ é a potência mínima que é possível gerar no barramento i;

– $pc_i$ é a potência consumida no barramento i;

– Smax é a potência disponível;

– Smin é a potência mínima disponível estando todos os barramentos com geração a produzir;

– Ped é a potência pedida pelas cargas;

– $pg_i$ é a potência produzida no barramento i;

– $\Delta pg_i$ é o acréscimo de potência gerada no barramento i acima da potência mínima gerada desse barramento.

Atendendo a que a soma das potências consumidas é igual à soma das potências geradas, temos:

$$Ped = \sum ( min_i + \Delta pg_i )$$

Donde se conclui que:

$$\sum \Delta pg_i = Ped - Smin$$

No critério de proporcionalidade, cada $\Delta pg_i$ vai ser proporcional à diferença $max_i - min_i$. Portanto, tem-se que:

$$\Delta pg_i = \frac{max_i - min_i}{\Sigma ( max_i - min_i )} ( Ped - Smin ) \qquad (5.6)$$

Atendendo a 5.5 e a 5.6, tem-se:

$$pg_i = min_i + \frac{max_i - min_i}{Smax - Smin} ( Ped - Smin ) \qquad (5.7)$$

Desenvolvendo a expressão 5.7, obtém-se finalmente a expressão que nos dá o despacho inicial em cada barramento:

$$pg_i = \frac{min_i ( Smax - Ped ) + max_i ( Ped - Smin )}{Smax - Smin} \qquad (5.8)$$

Desta forma, começa-se por fazer a análise de um estado com um despacho equilibrado por todos os barramentos de produção, o que conduzirá a uma repartição mais equitativa dos fluxos nos ramos, reduzindo-se assim o aparecimento de violações.

## 5.4 – Correcção do fluxo num ramo

Quando há violação da capacidade dum ramo, acontece um dos seguintes casos:

– O fluxo é positivo e superior à capacidade do ramo (violação positiva);

– O fluxo é negativo e inferior ao simétrico da capacidade do ramo (violação negativa).

De forma a corrigir a violação, o fluxo deve ser diminuído no primeiro caso e aumentado no segundo caso. Note-se que aumentar um fluxo negativo é aproximá-lo de zero, ou seja, diminuí-lo em módulo.

O fluxo no ramo r ($f_r$) é dado por [23]:

$$f_r = \sum_k a_{rk} \cdot p_k \qquad (5.9)$$

em que:

– $a_{rk}$ é o coeficiente de sensibilidade do ramo r relativamente ao barramento k;

– $p_k$ é a potência injectada no barramento k.

Seguidamente, vai-se determinar qual a variação do fluxo $f_r$ quando há um redespacho em que se transfere a potência gerada do barramento i para o barramento j. Como é suposto que as cargas se mantêm constantes, a variação da potência injectada é igual à variação da potência gerada. Portanto, o novo fluxo no ramo r ($nf_r$) é dado por:

$$nf_r = \sum_{\substack{k \neq i \\ k \neq j}} (a_{rk} \cdot p_k) + a_{ri}(p_i - \Delta p) + a_{rj}(p_j + \Delta p) \qquad (5.10)$$

em que $\Delta p$ é positivo e representa a potência gerada que foi transferida do barramento i para o barramento j.

Conjugando as expressões 5.9 e 5.10, consegue-se calcular a variação de fluxo no ramo r ($\Delta f_r$) devido à referida transferência de potências geradas. Assim, tem-se:

$$\Delta f_r = nf_r - nf_r = \Delta p\,(a_{rj} - a_{ri}) \qquad (5.11)$$

Da expressão 5.11, vê-se que:

$$\Delta f_r > 0 \text{ se } a_{ri} < a_{rj} \qquad (5.12)$$

$$\Delta f_r < 0 \text{ se } a_{ri} > a_{rj} \qquad (5.13)$$

Da expressão 5.12, pode-se concluir que o fluxo num ramo aumenta quando se transfere potência gerada dos barramentos de menor coeficiente de sensibilidade para os barramentos de maior coeficiente de sensibilidade. Desta forma, é possível tentar corrigir uma violação negativa de fluxo.

Da expressão 5.13 conclui-se que o fluxo num ramo diminui quando é transferida potência gerada dos barramentos de maior coeficiente de sensibilidade para os barramentos de menor coeficiente de sensibilidade. Desta forma, é possível tentar corrigir uma violação positiva de fluxo.

## 5.5 – Gráficos

O programa, para além de calcular os índices de fiabilidade já referidos, vai ter ainda a possibilidade de fornecer, para cada barramento, um gráfico de barras e um histograma.

O gráfico de barras relaciona o número de avarias anuais com a respectiva probabilidade. Para isso, é necessário obter a frequência (número de anos) com que ocorre cada número avarias anuais. Seguidamente, calcula-se a probabilidade de ocorrer um desses números de avarias anuais dividindo a respectiva frequência de ocorrência pelo número total de anos.

O histograma relaciona classes de indisponibilidade anual com a respectiva probabilidade. As classes de indisponibilidade anual têm uma amplitude de dez horas e são identificadas pelo seu valor médio, como exemplificamos a seguir:

– Classe 5 engloba todas as indisponibilidades anuais de 0h inclusive a 10h exclusive;

– Classe 15 engloba todas as indisponibilidades anuais de 10h inclusive a 20h exclusive;

e assim sucessivamente.

Tal como anteriormente, é necessário obter a frequência (número de anos) com que ocorre cada uma das classes de

indisponibilidade anual. Em seguida, calcula-se a probabilidade de ocorrer cada uma dessas classes dividindo a respectiva frequência de ocorrência pelo número total de anos.

## 5.6 – Estrutura do ficheiro de entrada de dados

No ficheiro de texto que é a entrada do programa, os dados devem figurar pela seguinte ordem:

– Número de barramentos;

– Potência de base (MVA) do sistema P.U. a utilizar;

– Várias linhas, contendo cada uma as seguintes informações correspondentes a um gerador:

– Barramento a que o gerador está ligado;

– Potência máxima (MW);

– Potência mínima (percentagem da potência máxima);

– Tempo médio de funcionamento (horas);

– Tempo médio de reparação (horas);

– Uma linha com um zero que indica o local onde terminam as linhas correspondentes aos geradores;

– Várias linhas, contendo cada uma as seguintes informações correspondentes a um ramo:

– Barramentos a que o ramo está ligado;

– Capacidade de transmissão (MVA);

– Reactância (P.U.);

– Taxa de avarias ($ano^{-1}$);

– Tempo médio de reparação (horas);

– Uma linha com um zero que indica o local onde terminam as linhas correspondentes aos ramos;

– Várias linhas, contendo cada uma as seguintes informações correspondentes a uma carga:

– Barramentos a que a carga está ligada;

– Prioridade de deslastre;

– Pico anual (MW);

– Uma linha com um zero que indica o local onde terminam as linhas correspondentes às cargas;

– Picos semanais das cargas como percentagens dos respectivos picos anuais (52 percentagens);

– Picos diários das cargas como percentagens dos respectivos picos semanais (7 percentagens).

## 5.7 – Estruturas de dados do programa

As principais estruturas de dados utilizadas no programa são as seguintes:

– Variável nb que representa o número de barramentos do sistema;

– Variável sb que representa a potência de base do sistema P.U. a utilizar;

– Vector g, de ng elementos, onde se armazena as informações relativas aos geradores e contendo cada elemento os seguintes campos:

– bar → Barramento a que o gerador está ligado;

– max → Potência máxima do gerador;

– min → Potência mínima do gerador;

– Vector tx → Vector de dois elementos onde são armazenadas as taxas de avaria e de reparação;

– e → Estado em que o gerador se encontra (0 – funcionamento, 1 – avaria);

– Vector r, de nr elementos, onde se armazena as informações relativas aos ramos e contendo cada elemento os seguintes campos:

– bar1 e bar2 → Barramentos a que o ramo está ligado;

– max → Capacidade de transmissão do ramo;

– x → Reactância do ramo;

– Vector tx → Vector de dois elementos onde são armazenadas as taxas de avaria e de reparação;

– e → Estado em que o ramo se encontra (0 – funcionamento, 1 – avaria);

– Vector c, de nc elementos, onde se armazena as informações relativas às cargas e contendo cada elemento os seguintes campos:

– bar → Barramentos a que a carga está ligada;

– prior → Prioridade de deslastre da carga;

– pico → Pico anual da carga;

– Vector av → Vector da frequência do número de avarias anuais;

– Vector u → Vector da frequência das classes de indisponibilidade anual;

– ava → Número de avarias durante o ano que está a ser simulado;

– avtot → Número total de avarias durante todos os anos de simulação;

– ua → Indisponibilidade durante o ano que está a ser simulado;

– utot → Indisponibilidade durante todos os anos de simulação;

– e → Estado em que a carga se encontra (0 – funcionamento, 1 – avaria);

– Vector sm, onde se armazena 52 percentagens do pico anual, correspondendo cada uma ao respectivo pico semanal;

– Vector dia, onde se armazena 7 percentagens do pico semanal, correspondendo cada uma ao respectivo pico diário;

– Matriz bi que corresponde à matriz B invertida;

– Matriz a que corresponde à matriz dos coeficientes de sensibilidade;

– Vector bar, de nb elementos, onde se armazena as informações relativas aos barramentos e contendo cada elemento os seguintes campos:

– max → Potência máxima disponível no barramento;

– min → Potência mínima do barramento se este estiver a produzir;

– pg → Potência gerada no barramento;

– pc → Potência consumida no barramento;

– pi → Potência injectada pelo barramento na rede;

– Vector bg de nbg elementos onde se armazena os números dos barramentos com geradores disponíveis;

– Variável ks que indica o componente sorteado (gerador ou ramo);

– Variável dt que indica o tempo de duração dum estado do sistema (em anos);

– Variável t que indica o tempo ao longo dum ano (em anos);

– Vector cs, de nbg elementos, onde se armazena números de barramentos com geração e os coeficientes de sensibilidade dum ramo relativamente a esses barramentos. Daí que o comprimento do vector cs seja de nbg elementos. Assim, cada elemento contém os seguintes campos:

– bar → Barramento (com geração) a que se refere o coeficiente de sensibilidade;

– coef → Coeficiente de sensibilidade.

## 5.8 – Fluxogramas do programa

Como o programa foi desenvolvido em linguagem C e como nesta linguagem os índices dos vectores começam em zero, então foi conveniente que o programa ajustasse a numeração dos barramentos, subtraindo-lhe uma unidade. Assim, em termos de programação, o barramento 1 passa a ser identificado pelo número zero, o barramento 2 passa a ser identificado pelo número um e assim sucessivamente. Daí que nos fluxogramas que se seguem, a numeração dos barramentos comece em zero.

Dada a complexidade e extensão do programa, são apresentados apenas os fluxogramas fundamentais para a compreensão do programa. De qualquer forma, teremos no anexo E o programa completo.

A fig. 5.3, apresenta o fluxograma do módulo principal. Durante a simulação, são armazenados dados (no vector c) que, mais tarde, vão permitir obter o cálculo dos resultados pretendidos.

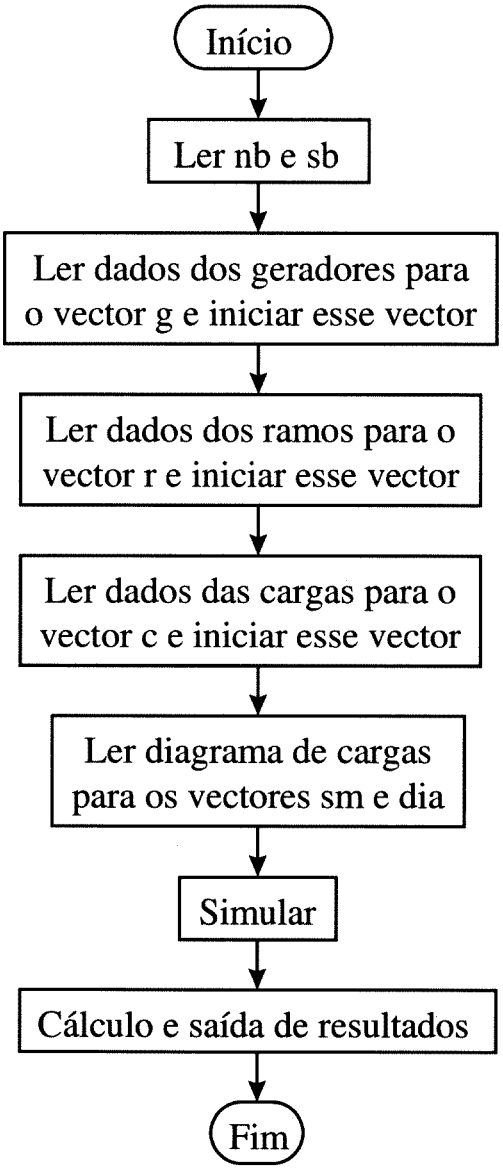



**fig. 5.3 – Módulo principal do programa**

Na fig. 5.4, está o fluxograma do módulo de simulação.

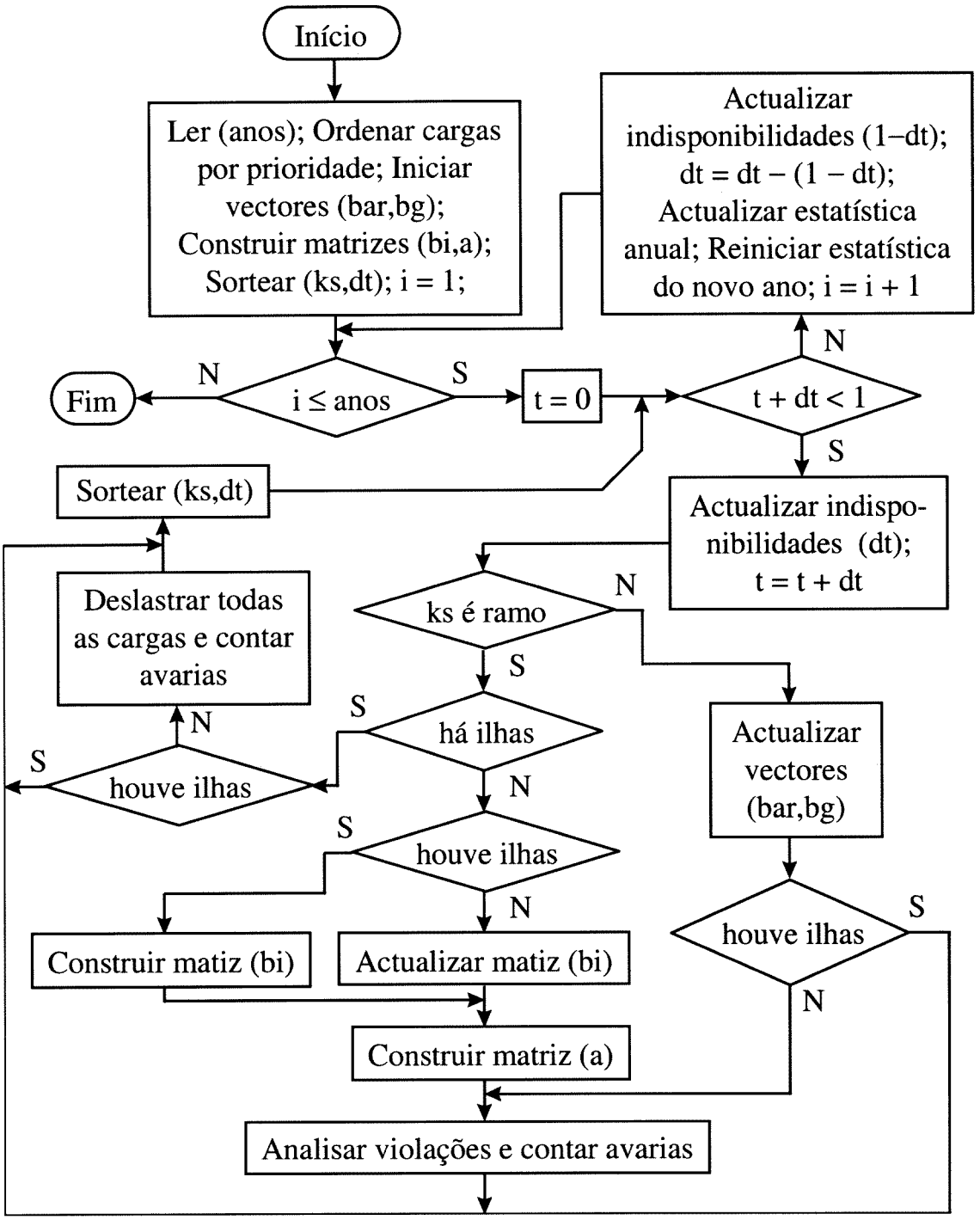



**fig. 5.4 – Módulo de simulação**

A fig. 5.5, representa o fluxograma que faz o sorteio da duração do estado actual do sistema e do próximo componente (gerador ou ramo) a mudar de estado. A justificação do processo de realização desses sorteios [6] já foi descrito no parágrafo 3.7.3.

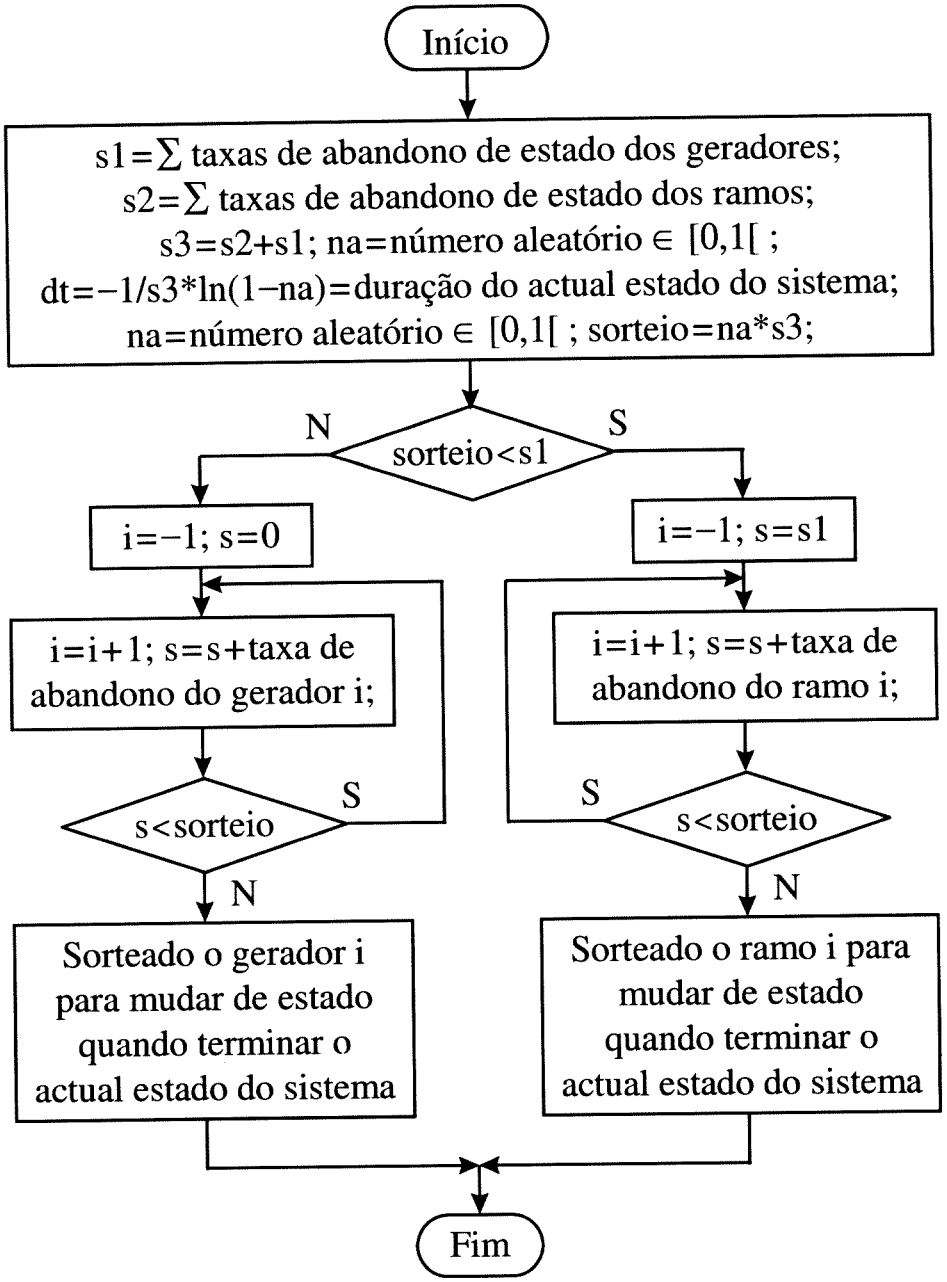



**fig. 5.5 – Sorteio de duração do estado actual do sistema e do próximo componente a mudar de estado**

A matriz B invertida do estado actual só é obtida a partir da matriz homóloga do estado anterior, se este não tiver sido um estado de "ilhas". Quando é efectuada a análise de estado e contagem de avarias, o tempo t determina o pico diário no momento da avaria, segundo o diagrama de cargas fornecido. O vector bg serve para indicar quais os barramentos onde se pode actuar quando fazemos uma correcção de fluxo através de redespacho, ou seja, os barramentos com produção. O vector bar contém toda a informação dos barramentos necessária para

efectuar redespachos e, consequentemente, corrigir as violações de fluxo.

Seguidamente, são apresentados os fluxogramas das partes mais importantes que integram o módulo simular. Na fig. 5.6, apresenta-se o fluxograma da construção da matriz B invertida.

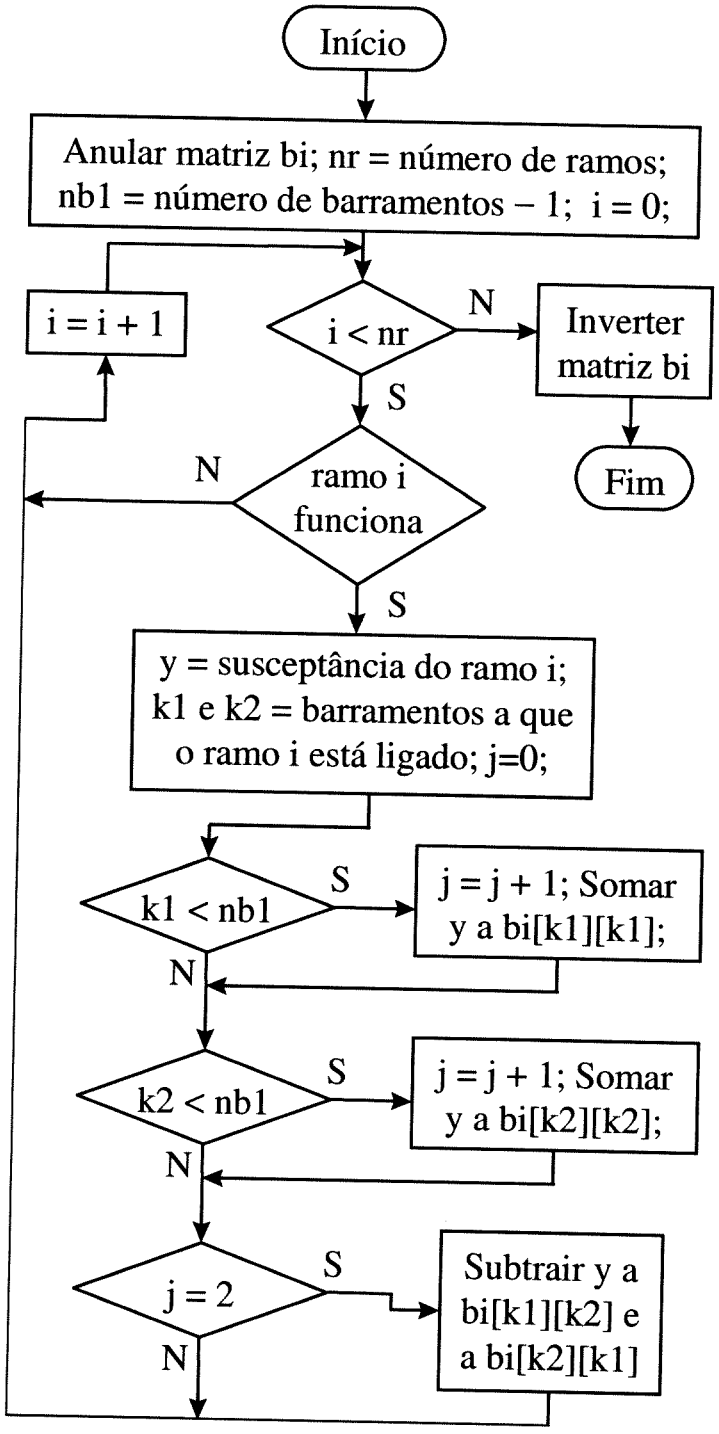



**fig. 5.6 – Construção da matriz B invertida**

Genericamente, a inversão duma matriz é obtida seguindo as regras seguintes [7]:

(1) $A'_{ii} = 1/A_{ii}$

(2) $A'_{ji} = A_{ji} A'_{ii}$ para todo $j \neq i$

(3) $A'_{jk} = A_{jk} - A'_{ji} A_{ik}$ para todo $j \neq i$ e $k \neq i$

(4) $A'_{ik} = - A'_{ii} A_{ik}$ para todo $k \neq i$

em que a matriz A é matriz original e A' é a matriz invertida.

Na fig. 5.7, pode ser visto o fluxograma da actualização da matriz B invertida a partir da matriz homóloga anterior. O processo usado [20] já foi descrito no parágrafo 5.2 e as expressões de cálculo dos elementos dos vectores auxiliares x1 e x2 resulta da resolução das expressões 5.3 e 5.4. Como já foi referido, só é possível aplicar este método se não houver "ilhas".

O valor da variável x é positivo se o elemento entrou em funcionamento e negativo na situação oposta. Em qualquer das situações, o valor absoluto de x é sempre igual à reactância do ramo sorteado.

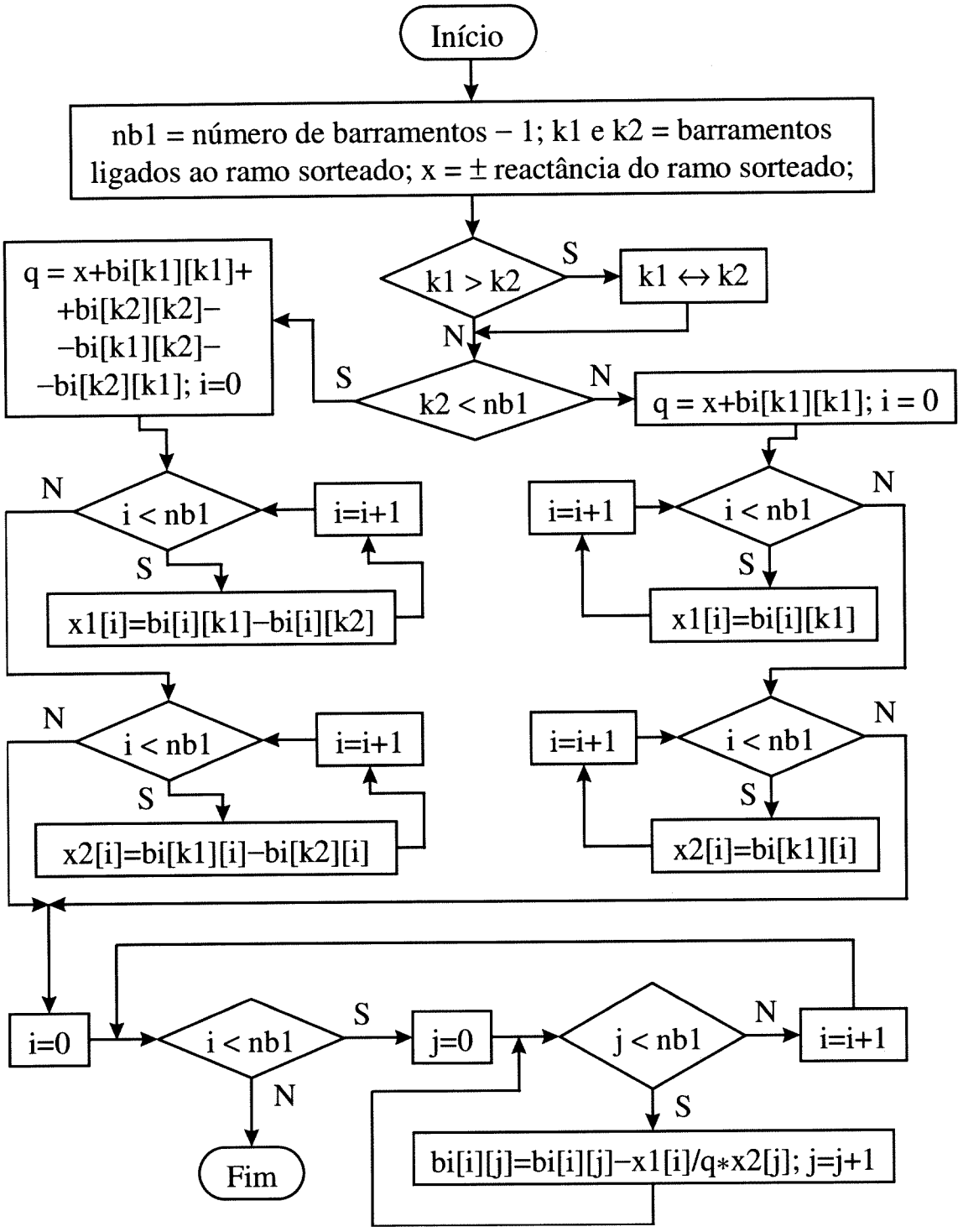



**fig. 5.7 – Actualização da matriz B invertida**

Na fig. 5.8, está o fluxograma da construção da matriz das sensibilidades a partir da matriz B invertida [23]. Para que isso fosse possível, foi acrescentado previamente à matriz B invertida uma linha nula, correspondente ao barramento de referência.

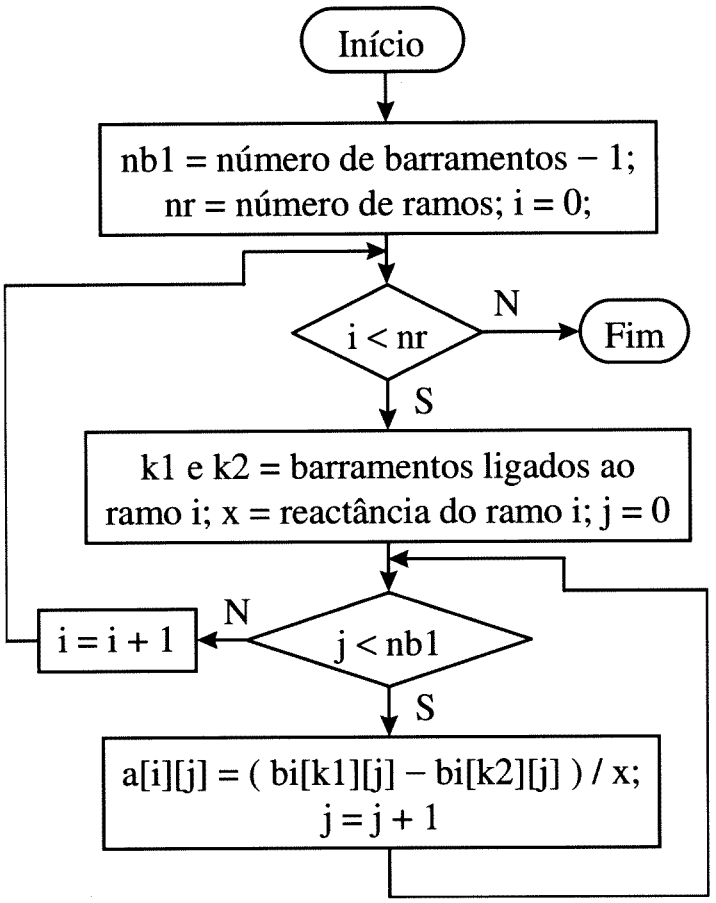



**fig. 5.8 – Construção da matriz das sensibilidades**

Na fig. 5.9, está representado o fluxograma da actualização do vector dos barramentos (vector bar) e do vector que indica os barramentos com geração (vector bg).

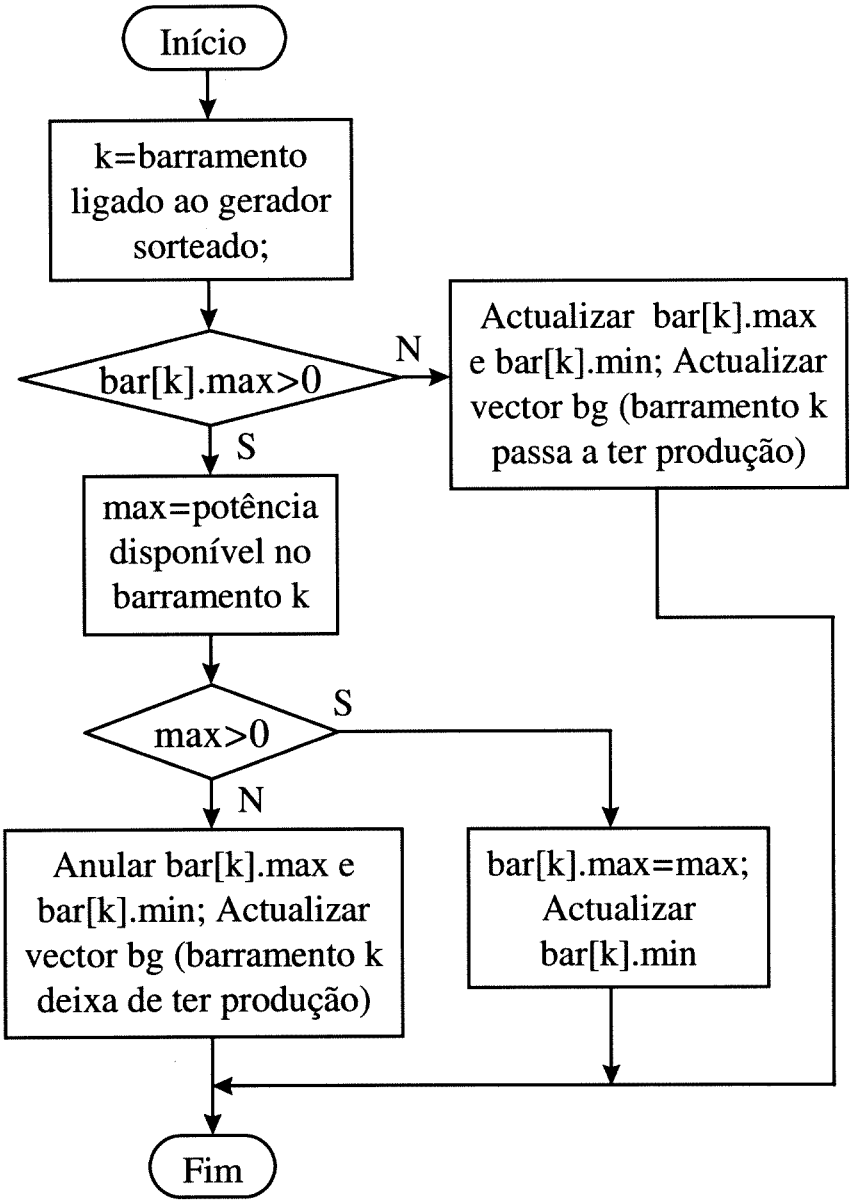



**fig. 5.9 – Actualização dos vectores bar e bg**

A fig. 5.10, mostra o fluxograma que faz a análise das violações e a contagem de avarias nos barramentos deslastrados.

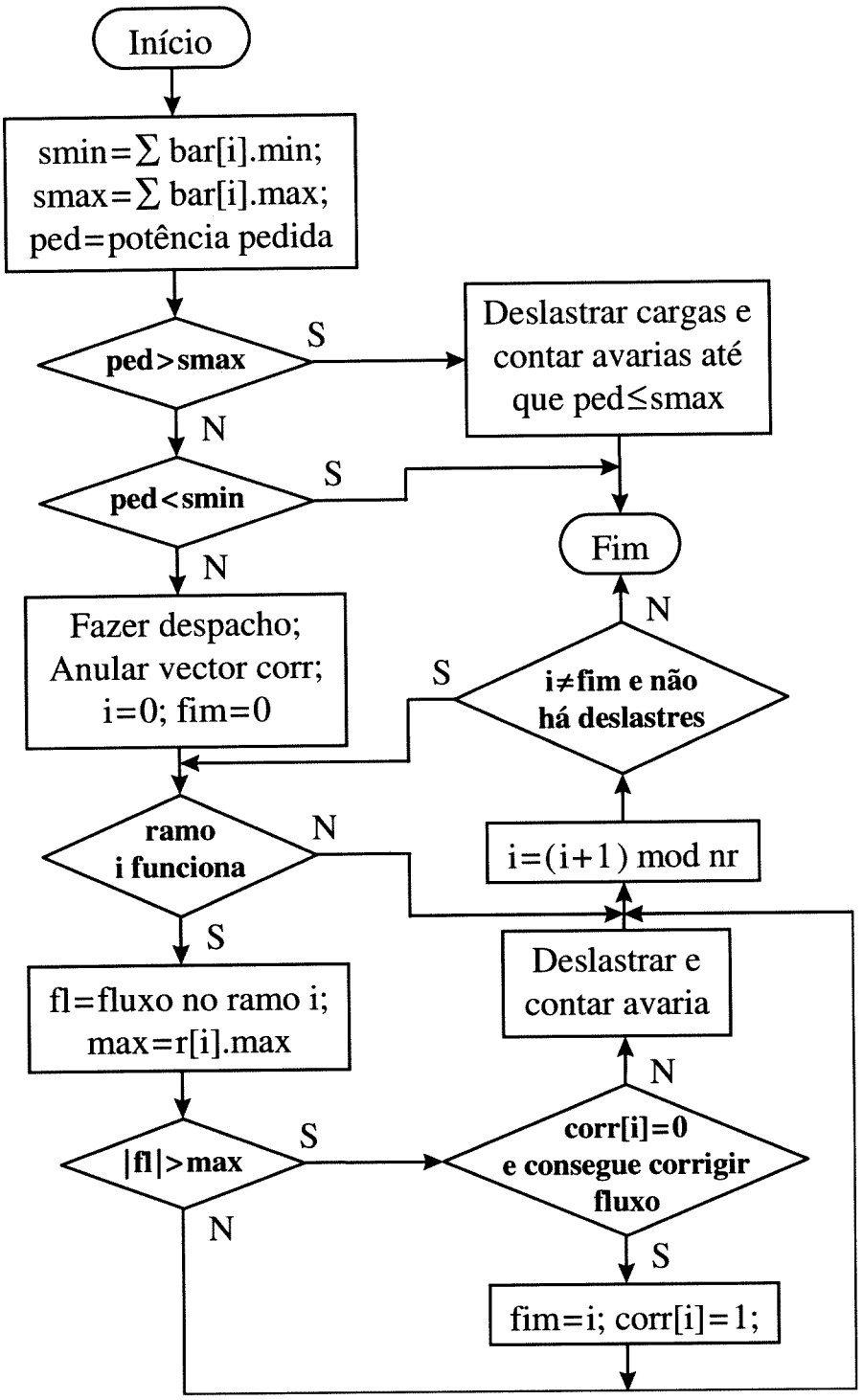



**fig. 5.10 – Análise de violações e contagem de avarias**

Na fig. 5.11, pode-se ver o fluxograma que tenta corrigir o fluxo num ramo.

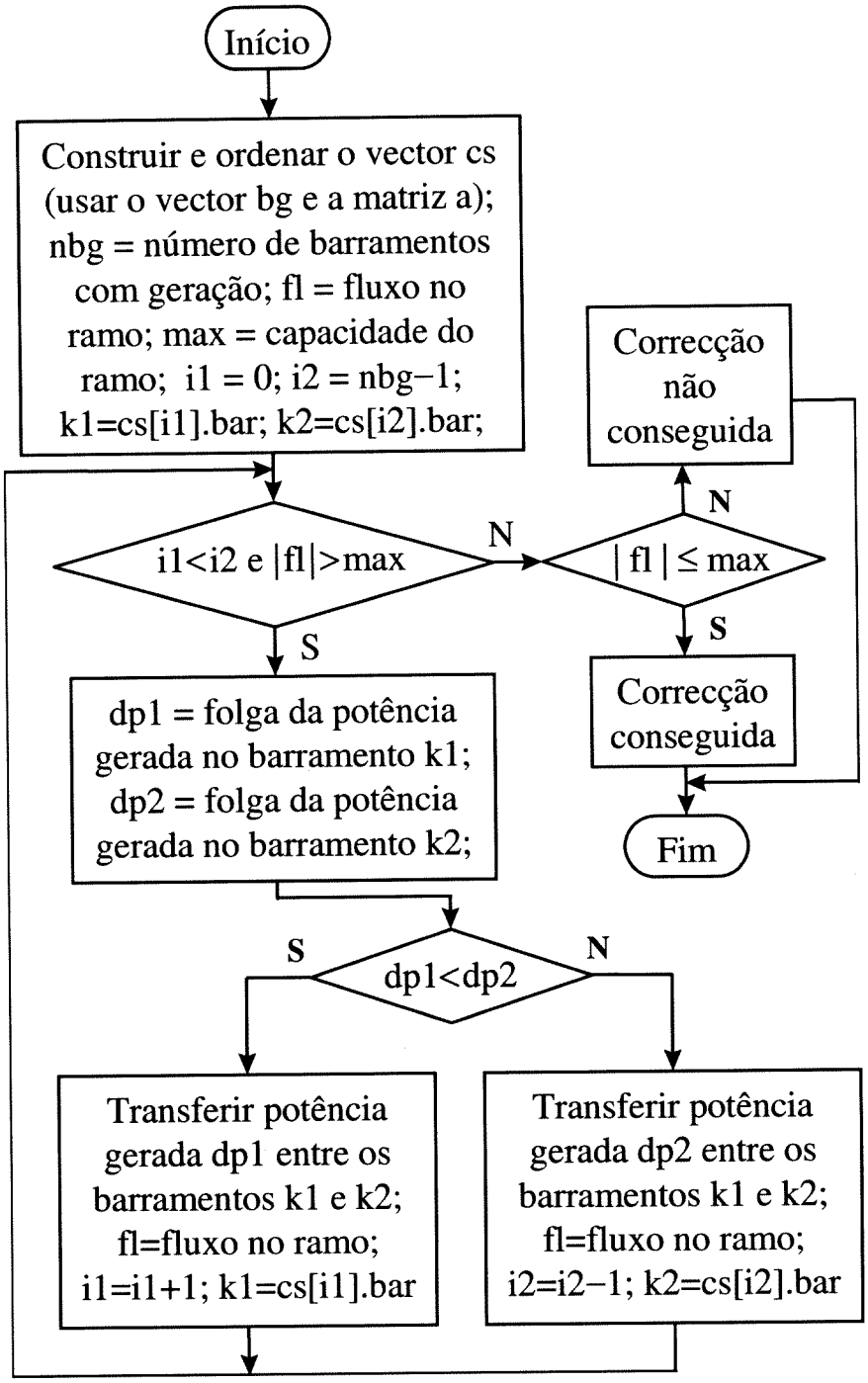



**fig. 5.11 – Correcção de fluxo num ramo**

Na fig. 5.12, é ,apresentado o fluxograma que verifica se há "ilhas". A variável ni conta o número de "ilhas" que vão surgindo e n indica o número de barramentos que ainda não estão ligados a nenhuma ilha. No final, se não houver "ilhas", a variável ni deve ser igual a um, o que significa que o sistema só possui uma ilha que é a própria rede, quando está conexa e a variável n deve ser zero, o que significa que todos os barramento foram ligados à rede. Se alguma destas condições

não se verificar então há "ilhas", podendo uma destas ser constituída apenas por um barramento isolado.

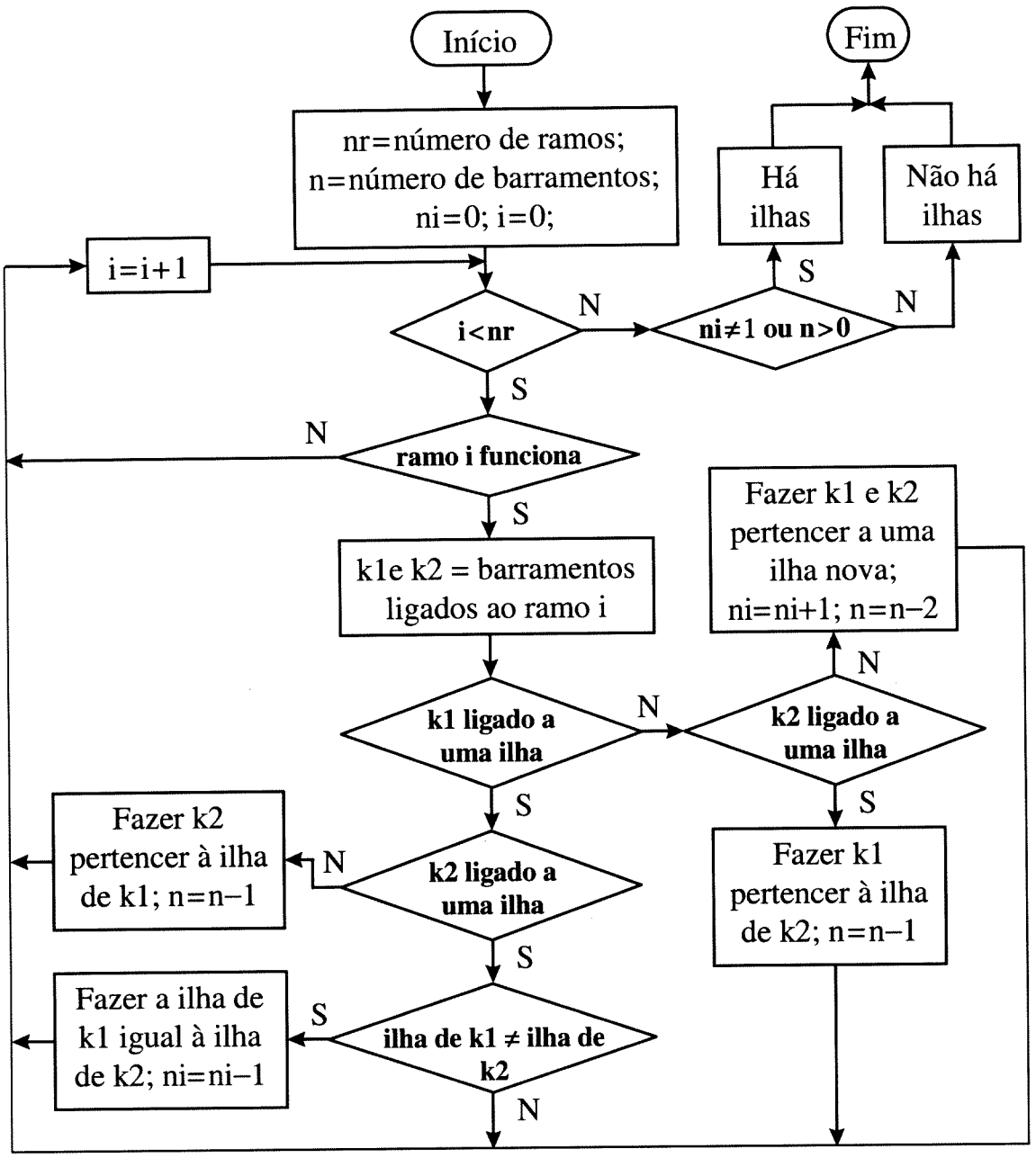



**fig. 5.12 – Verificação de existência de "ilhas"**

Na fig. 5.13, vê-se o fluxograma do módulo que calcula os resultados e efectua a respectiva saída. O utilizador pode escolher a saída dentro de um menu com as seguintes opções:

– Visualizar uma tabela com os índices de fiabilidade em cada barramento;

– Visualizar para cada barramento uma tabela com as probabilidades de ocorrer um determinado número de avarias por ano e outra com as probabilidades de ocorrer uma determinada classe de indisponibilidade;

– Visualizar graficamente as tabelas da opção anterior;

– Gravar todos os resultados num ficheiro.

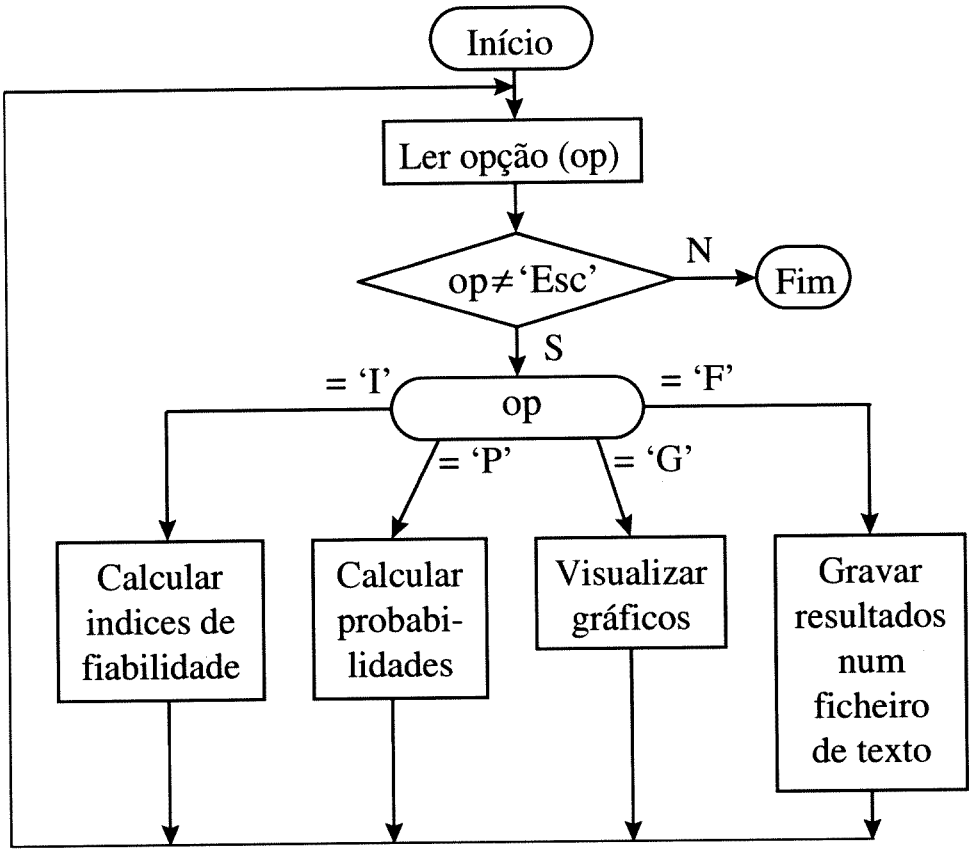



**fig. 5.13 – Cálculo e saída de resultados**

## 5.9 – Sistema para testar o programa

Para testar o programa, vai ser usado o sistema de teste de fiabilidade IEEE [11]. Na fig. 5.14, observa-se esse sistema.

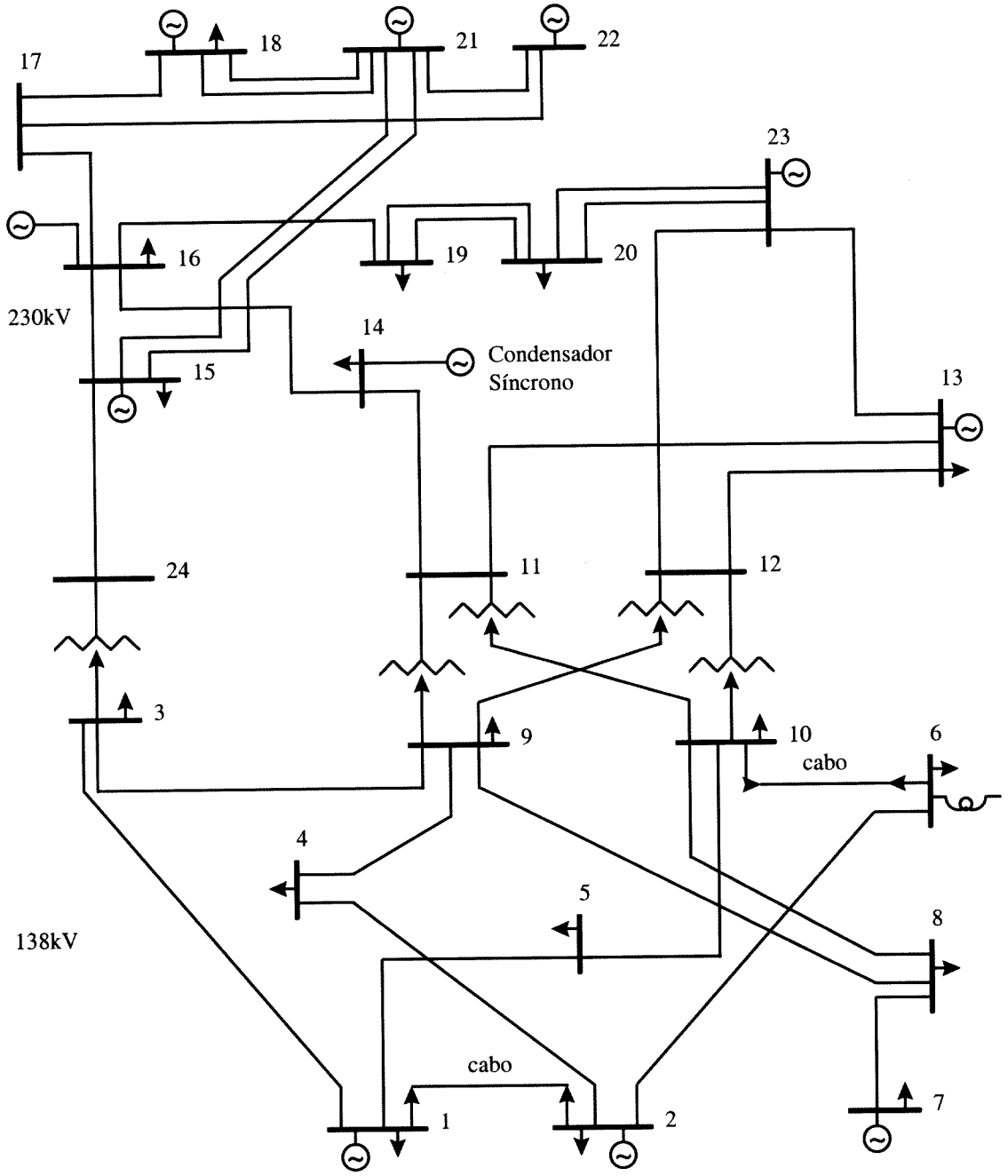



**fig. 5.14 – Sistema de teste de fiabilidade IEEE**

O número de barramentos deste sistema é 24 e a potência de base adoptada é 100MVA.

Na tabela 5.1, estão os dados dos geradores:

| Dados dos geradores | | | | |
|---|---|---|---|---|
| **Bar** | **Max (MW)** | **Min (%)** | **m (h)** | **r (h)** |
| 1 | 20 | 80 | 450 | 50 |
| 1 | 20 | 80 | 450 | 50 |
| 1 | 76 | 20 | 1960 | 40 |
| 1 | 76 | 20 | 1960 | 40 |
| 2 | 20 | 80 | 450 | 50 |
| 2 | 20 | 80 | 450 | 50 |
| 2 | 76 | 20 | 1960 | 40 |
| 2 | 76 | 20 | 1960 | 40 |
| 7 | 100 | 25 | 1200 | 50 |
| 7 | 100 | 25 | 1200 | 50 |
| 7 | 100 | 25 | 1200 | 50 |
| 13 | 197 | 35 | 950 | 50 |
| 13 | 197 | 35 | 950 | 50 |
| 13 | 197 | 35 | 950 | 50 |
| 15 | 12 | 20 | 2940 | 60 |
| 15 | 12 | 20 | 2940 | 60 |
| 15 | 12 | 20 | 2940 | 60 |
| 15 | 12 | 20 | 2940 | 60 |
| 15 | 12 | 20 | 2940 | 60 |
| 15 | 155 | 35 | 960 | 40 |
| 16 | 155 | 35 | 960 | 40 |
| 18 | 400 | 25 | 1100 | 150 |
| 21 | 400 | 25 | 1100 | 150 |
| 22 | 50 | 90 | 1980 | 20 |
| 22 | 50 | 90 | 1980 | 20 |
| 22 | 50 | 90 | 1980 | 20 |
| 22 | 50 | 90 | 1980 | 20 |
| 22 | 50 | 90 | 1980 | 20 |
| 22 | 50 | 90 | 1980 | 20 |
| 23 | 155 | 35 | 960 | 40 |
| 23 | 155 | 35 | 960 | 40 |
| 23 | 350 | 40 | 1150 | 100 |

**Tabela 5.1 – Dados dos geradores**

Na tabela 5.2, são apresentados os dados dos ramos:

| Dados dos ramos | | | | | |
|---|---|---|---|---|---|
| **Bar 1** | **Bar 2** | **Max (MVA)** | **X (P. U.)** | **λ (av/ano)** | **r (h)** |
| 1 | 2 | 175 | 0,0139 | 0,24 | 16 |
| 1 | 3 | 175 | 0,2112 | 0,51 | 10 |
| 1 | 5 | 175 | 0,0845 | 0,33 | 10 |
| 2 | 4 | 175 | 0,1267 | 0,39 | 10 |
| 2 | 6 | 175 | 0,1920 | 0,48 | 10 |
| 3 | 9 | 175 | 0,1190 | 0,38 | 10 |
| 3 | 24 | 400 | 0,0839 | 0,02 | 768 |
| 4 | 9 | 175 | 0,1037 | 0,36 | 10 |
| 5 | 10 | 175 | 0,0883 | 0,34 | 10 |
| 6 | 10 | 175 | 0,0605 | 0,33 | 35 |
| 7 | 8 | 175 | 0,0614 | 0,30 | 10 |
| 8 | 9 | 175 | 0,1651 | 0,44 | 10 |
| 8 | 10 | 175 | 0,1651 | 0,44 | 10 |
| 9 | 11 | 400 | 0,0839 | 0,02 | 768 |
| 9 | 12 | 400 | 0,0839 | 0,02 | 768 |
| 10 | 11 | 400 | 0,0839 | 0,02 | 768 |
| 10 | 12 | 400 | 0,0839 | 0,02 | 768 |
| 11 | 13 | 500 | 0,0476 | 0,40 | 11 |
| 11 | 14 | 500 | 0,0418 | 0,39 | 11 |
| 12 | 13 | 500 | 0,0476 | 0,40 | 11 |
| 12 | 23 | 500 | 0,0966 | 0,52 | 11 |
| 13 | 23 | 500 | 0,0865 | 0,49 | 11 |
| 14 | 16 | 500 | 0,0389 | 0,38 | 11 |
| 15 | 16 | 500 | 0,0173 | 0,33 | 11 |
| 15 | 21 | 500 | 0,0490 | 0,41 | 11 |
| 15 | 21 | 500 | 0,0490 | 0,41 | 11 |
| 15 | 24 | 500 | 0,0519 | 0,41 | 11 |
| 16 | 17 | 500 | 0,0259 | 0,35 | 11 |
| 16 | 19 | 500 | 0,0231 | 0,34 | 11 |
| 17 | 18 | 500 | 0,0144 | 0,32 | 11 |
| 17 | 22 | 500 | 0,1053 | 0,54 | 11 |
| 18 | 21 | 500 | 0,0259 | 0,35 | 11 |
| 18 | 21 | 500 | 0,0259 | 0,35 | 11 |
| 19 | 20 | 500 | 0,0396 | 0,38 | 11 |
| 19 | 20 | 500 | 0,0396 | 0,38 | 11 |
| 20 | 23 | 500 | 0,0216 | 0,34 | 11 |
| 20 | 23 | 500 | 0,0216 | 0,34 | 11 |
| 21 | 22 | 500 | 0,0678 | 0,45 | 11 |

**Tabela 5.2 – Dados dos ramos**

A tabela 5.3, indica os dados das cargas:

| Dados das cargas | | |
|---|---|---|
| **Bar** | **Prioridade** | **Pico anual (MW)** |
| 1 | 1 | 108 |
| 2 | 2 | 97 |
| 3 | 3 | 180 |
| 4 | 4 | 74 |
| 5 | 5 | 71 |
| 6 | 6 | 136 |
| 7 | 7 | 125 |
| 8 | 8 | 171 |
| 9 | 9 | 175 |
| 10 | 10 | 195 |
| 13 | 11 | 265 |
| 14 | 12 | 194 |
| 15 | 13 | 317 |
| 16 | 14 | 100 |
| 18 | 15 | 333 |
| 19 | 16 | 181 |
| 20 | 17 | 128 |

**Tabela 5.3 – Dados das cargas**

A tabela 5.4, apresenta os picos semanais em percentagem do respectivo pico anual:

| Picos semanais em percentagem do pico anual | | | |
|---|---|---|---|
| Semana | Pico (%) | Semana | Pico (%) |
| 1 | 86,2 | 27 | 75,5 |
| 2 | 90 | 28 | 81,6 |
| 3 | 87,8 | 29 | 80,1 |
| 4 | 83,4 | 30 | 88 |
| 5 | 88 | 31 | 72,2 |
| 6 | 84,1 | 32 | 77,6 |
| 7 | 83,2 | 33 | 80 |
| 8 | 80,6 | 34 | 72,9 |
| 9 | 74 | 35 | 72,6 |
| 10 | 73,7 | 36 | 70,5 |
| 11 | 71,5 | 37 | 78 |
| 12 | 72,7 | 38 | 69,5 |
| 13 | 70,4 | 39 | 72,4 |
| 14 | 75 | 40 | 72,4 |
| 15 | 72,1 | 41 | 74,3 |
| 16 | 80 | 42 | 74,4 |
| 17 | 75,4 | 43 | 80 |
| 18 | 83,7 | 44 | 88,1 |
| 19 | 87 | 45 | 88,5 |
| 20 | 88 | 46 | 90,9 |
| 21 | 85,6 | 47 | 94 |
| 22 | 81,1 | 48 | 89 |
| 23 | 90 | 49 | 94,2 |
| 24 | 88,7 | 50 | 97 |
| 25 | 89,6 | 51 | 100 |
| 26 | 86,1 | 52 | 95,2 |

**Tabela 5.4 – Picos semanais em percentagem do respectivo pico anual**

Na tabela 5.5, estão os picos diários em percentagem do respectivo pico semanal:

| Picos diários em percentagem do pico semanal | |
|---|---|
| Dia | Pico (%) |
| Segunda | 93 |
| Terça | 100 |
| Quarta | 98 |
| Quinta | 96 |
| Sexta | 94 |
| Sábado | 77 |
| Domingo | 75 |

**Tabela 5.5 – Picos diários em percentagem do respectivo pico semanal**

| | | Barramentos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9-10,13-16,18-20 |
| A | 0 | 25,4 | 46,4 | 54,7 | 67,8 | 71,0 | 71,8 | 73,2 | 73,6 | 74,1 |
| v | 1 | 30,0 | 33,5 | 31,3 | 25,8 | 23,1 | 23,2 | 22,7 | 22,5 | 22,0 |
| | 2 | 23,1 | 10,9 | 9,1 | 5,3 | 5,1 | 4,7 | 3,9 | 3,7 | 3,7 |
| p | 3 | 11,1 | 5,5 | 3,8 | 0,8 | 0,6 | 0,3 | 0,2 | 0,2 | 0,2 |
| o | 4 | 5,0 | 2,7 | 0,9 | 0,3 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 |
| r | 5 | 2,7 | 0,8 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | 6 | 1,5 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| a | 7 | 0,6 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| n | 8 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| o | 9 | 0,4 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | Resto | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |

**Tabela 5.7 – Probabilidades (em %) de ocorrência de avarias nos barramentos durante um ano**

Obteve-se ainda, para cada barramento, as probabilidades estimadas em percentagem de ocorrência de indisponibilidade anuais (agrupadas por classes de amplitude de dez horas), como se pode observar na tabela 5.8.

| | | Barramentos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9-10,13-16,18-20 |
| | 0-10 | 39,3 | 64,8 | 73,1 | 84,6 | 86,9 | 86,8 | 88,5 | 88,9 | 88,9 |
| I | 10-20 | 12,1 | 11,9 | 9,4 | 7,6 | 7,2 | 7,1 | 6,8 | 6,4 | 6,4 |
| n | 20-30 | 8,9 | 6,8 | 4,7 | 3,2 | 2,7 | 2,9 | 2,5 | 2,5 | 2,5 |
| d | 30-40 | 8,2 | 4,7 | 4,5 | 2,4 | 1,8 | 1,9 | 1,5 | 1,5 | 1,5 |
| i | 40-50 | 5,4 | 3,5 | 2,5 | 0,6 | 0,7 | 0,5 | 0,5 | 0,5 | 0,5 |
| s | 50-60 | 3,9 | 1,6 | 1,5 | 0,2 | 0,3 | 0,3 | 0,1 | 0,1 | 0,1 |
| p | 60-70 | 3,8 | 1,6 | 1,4 | 0,4 | 0,2 | 0,2 | 0,0 | 0,0 | 0,0 |
| o | 70-80 | 3,0 | 1,1 | 0,8 | 0,3 | 0,1 | 0,1 | 0,1 | 0,1 | 0,1 |
| n | 80-90 | 2,7 | 1,2 | 0,7 | 0,2 | 0,1 | 0,1 | 0,0 | 0,0 | 0,0 |
| i | 90-100 | 2,6 | 0,5 | 0,3 | 0,3 | 0,0 | 0,1 | 0,0 | 0,0 | 0,0 |
| b | 100-110 | 2,0 | 0,7 | 0,5 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| i | 110-120 | 1,2 | 0,6 | 0,2 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| l | 120-130 | 1,1 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| i | 130-140 | 1,6 | 0,2 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| d | 140-150 | 0,8 | 0,0 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| a | 150-160 | 0,2 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| d | 160-170 | 0,5 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| e | 170-180 | 0,7 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| s | 180-190 | 0,7 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | 190-200 | 0,2 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | Resto | 1,1 | 0,1 | 0,1 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |

**Tabela 5.8 – Probabilidades (em %) de ocorrência das indisponibilidades nos barramentos durante um ano**

Finalmente, o programa ainda dá a possibilidade de visualizar, para cada barramento, um gráfico de barras e um histograma, tal como se pode observar, desde a fig. 5.15 até à fig. 5.23. Nos histogramas das indisponibilidades, as classes são identificadas pelo respectivo valor médio, como já foi referido no parágrafo 5.5.

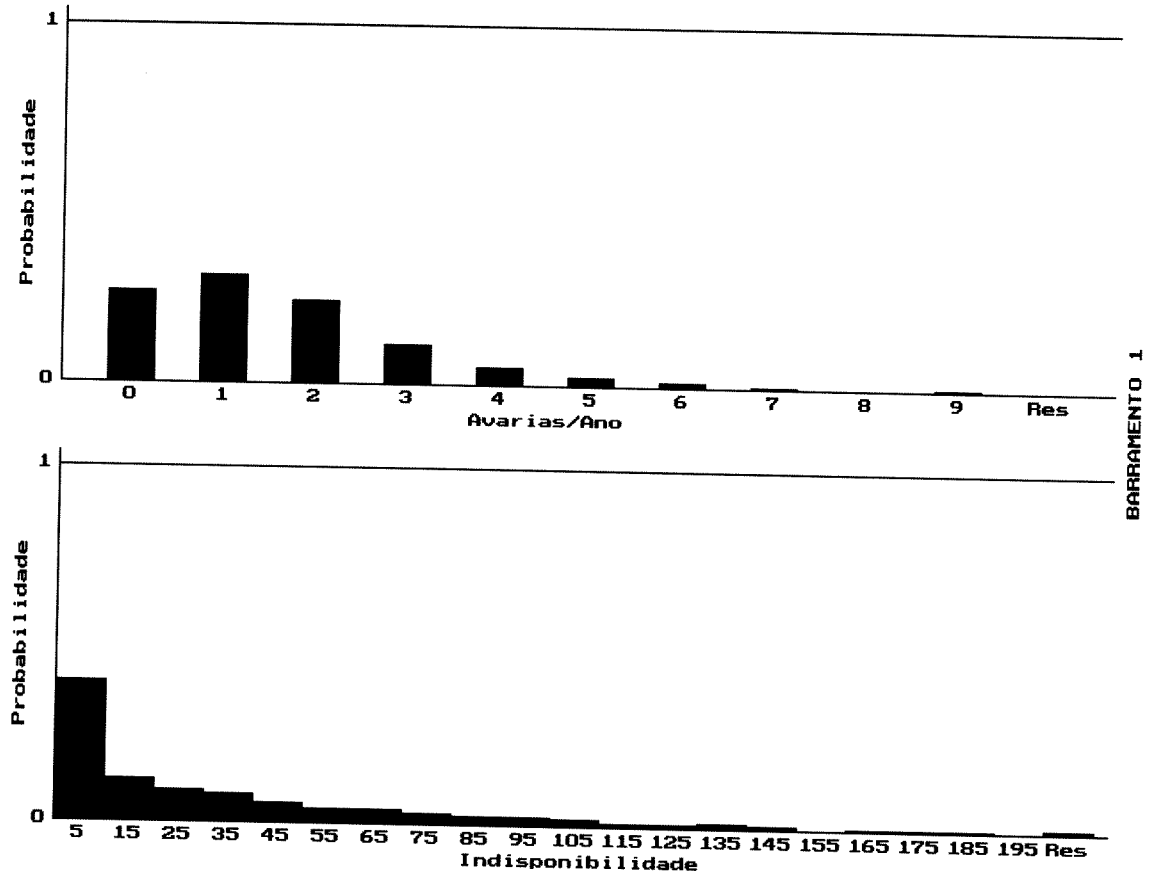



**fig. 5.15 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 1 durante um ano**

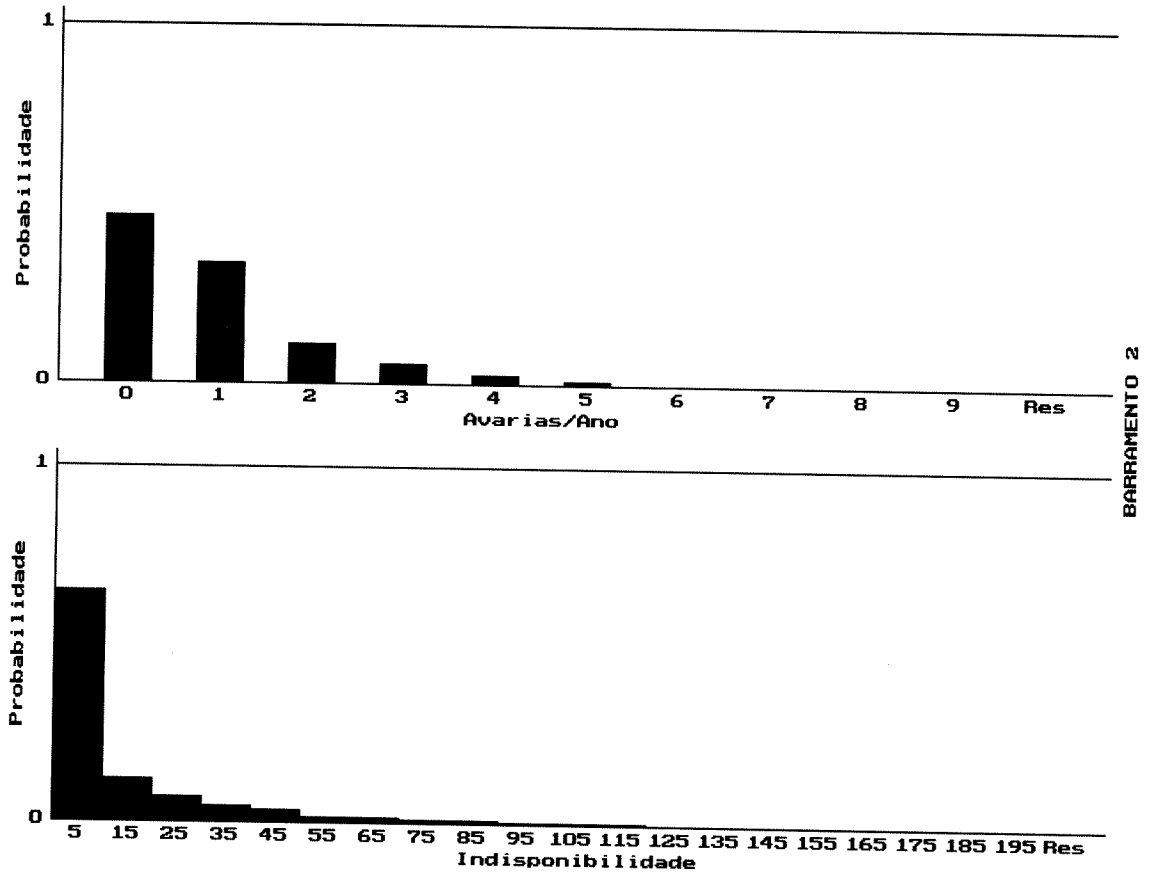



**fig. 5.16 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 2 durante um ano**

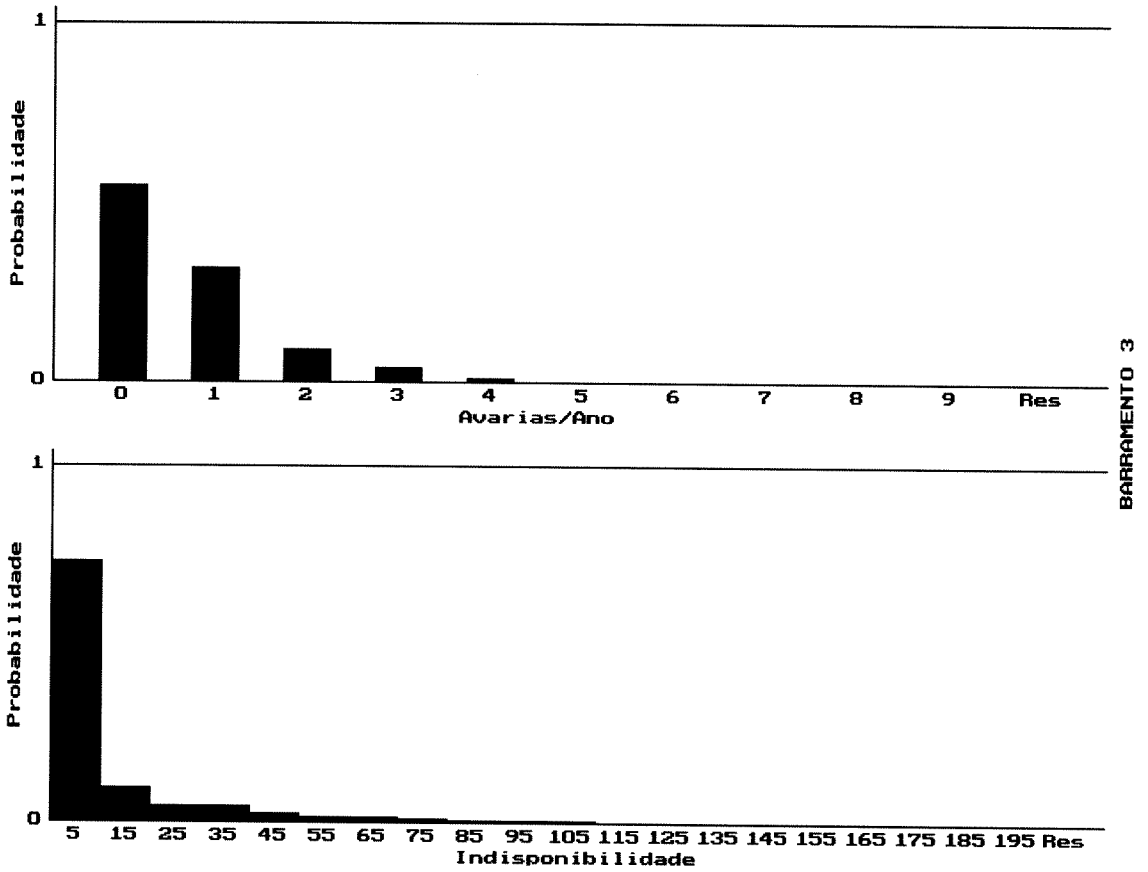



**fig. 5.17 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 3 durante um ano**

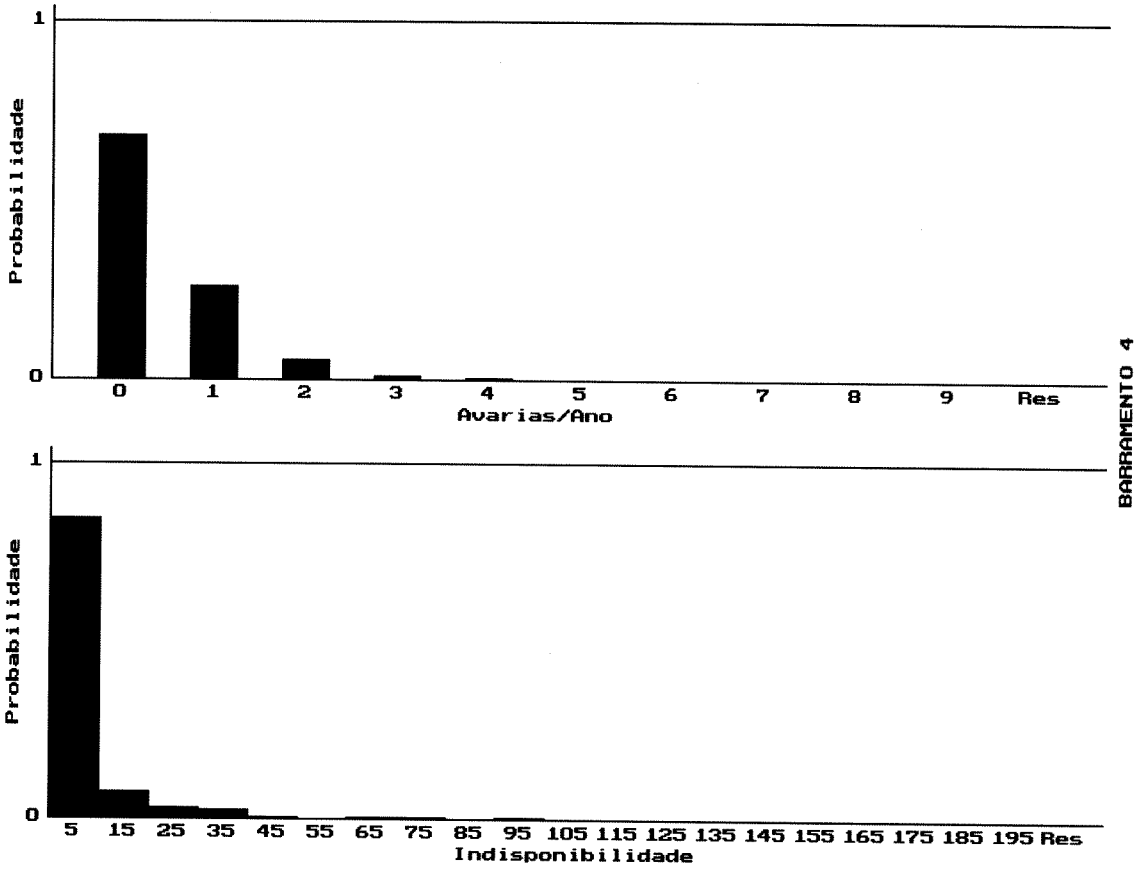



**fig. 5.18 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 4 durante um ano**

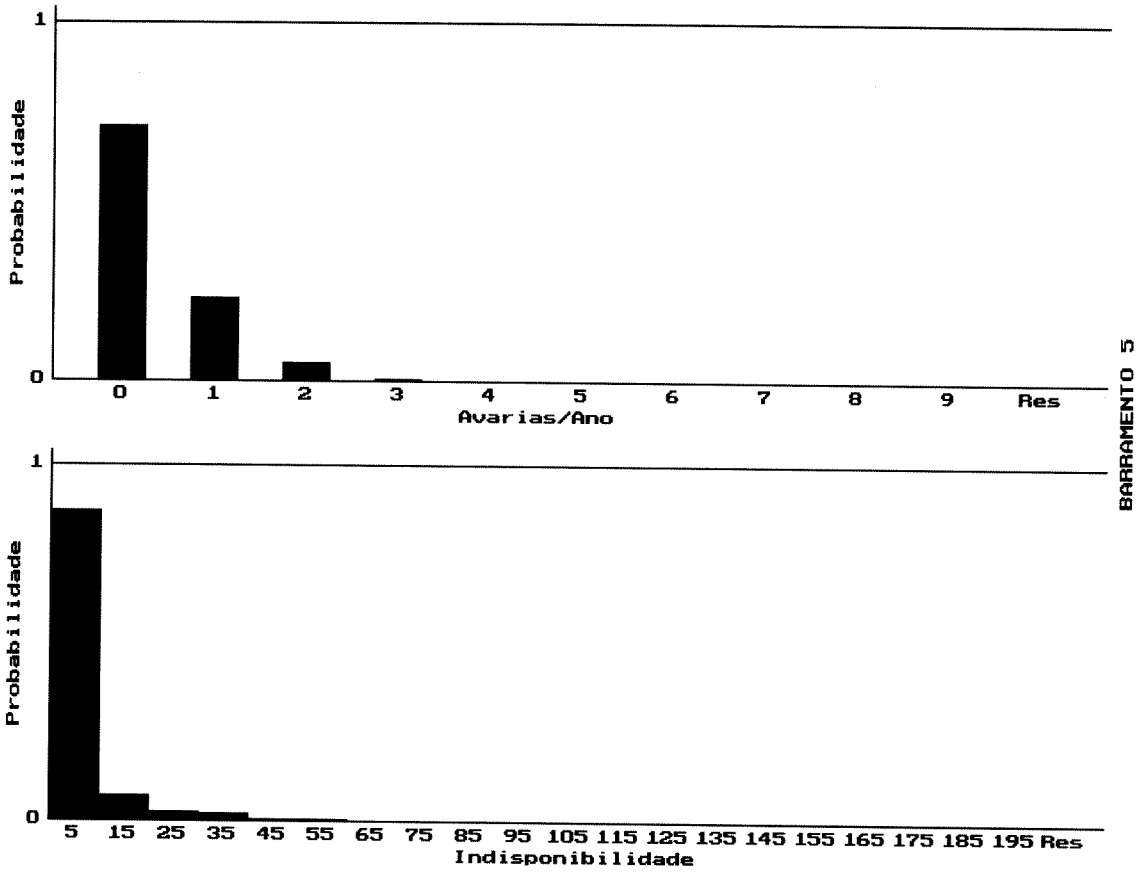



**fig. 5.19 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 5 durante um ano**

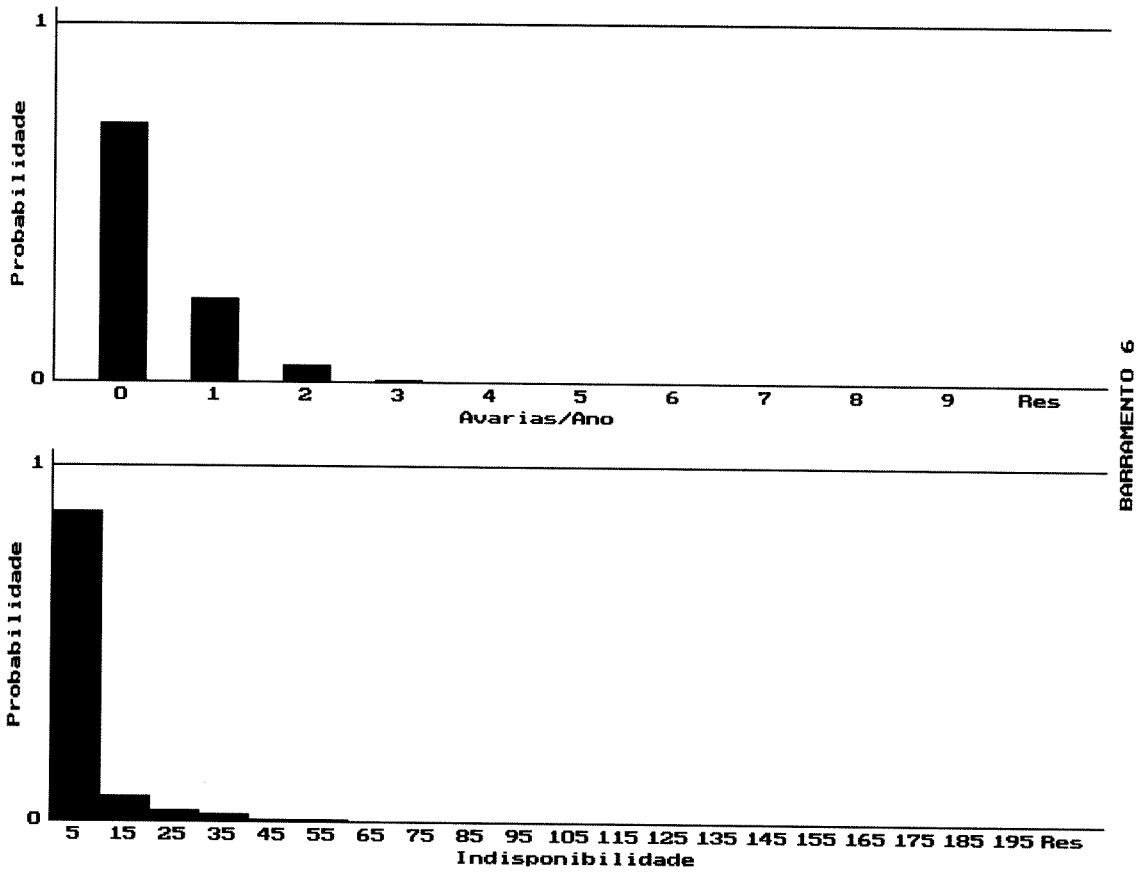



**fig. 5.20 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 6 durante um ano**

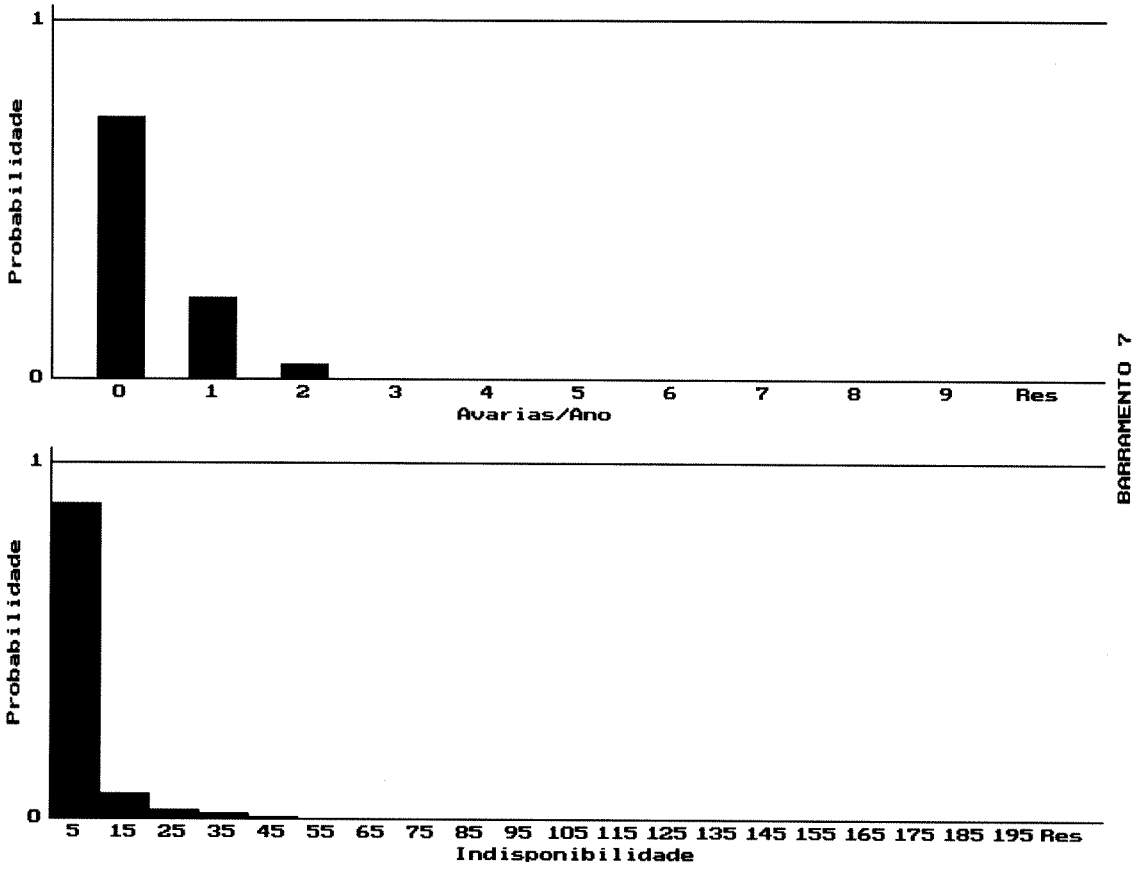



**fig. 5.21 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 7 durante um ano**

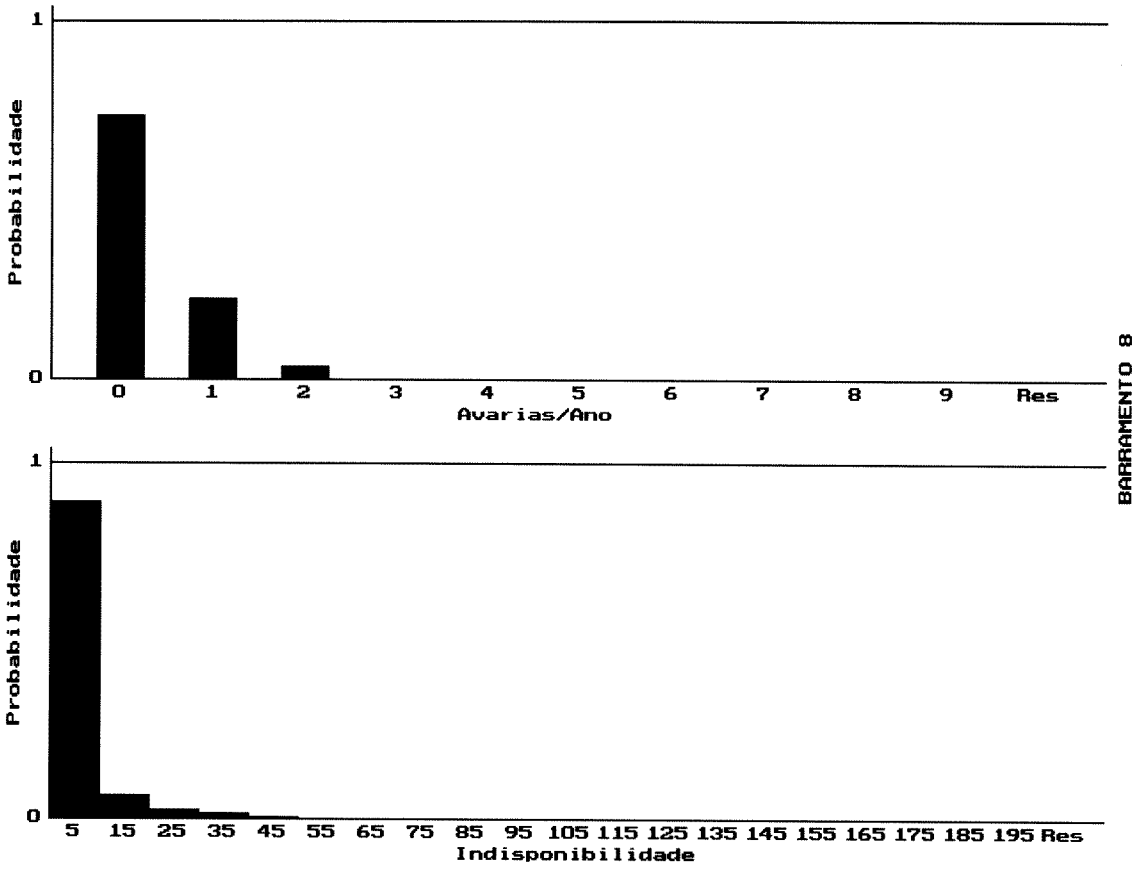



**fig. 5.22 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 8 durante um ano**

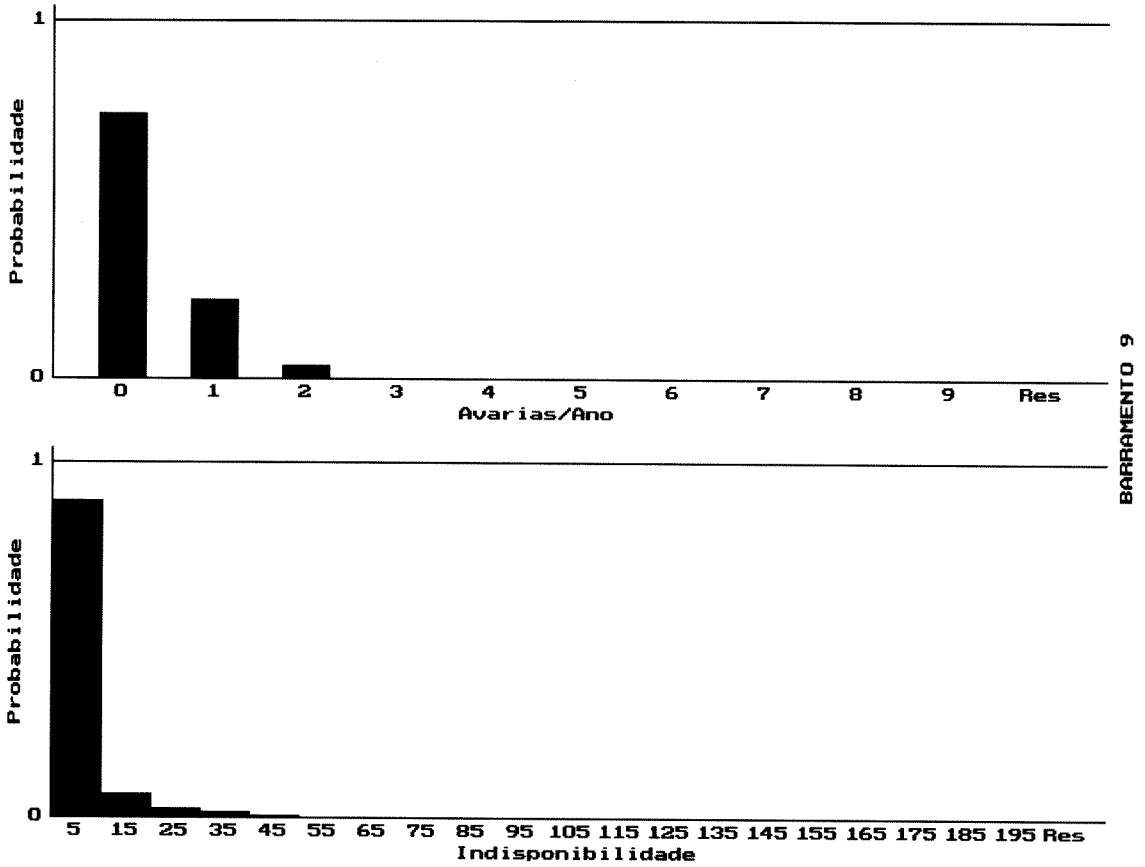



**fig. 5.23 – Gráficos das probabilidades das avarias e das indisponibilidades no barramento 9 durante um ano**

Para os barramentos 10, 13 a 16 e 18 a 20, tanto o gráfico de barras como o histograma são iguais aos da fig. 5.23.

## 5.11 – Conclusões

Neste capítulo, foi analisada a Fiabilidade do Sistema Composto utilizando o Método de Monte Carlo, combinado com o modelo DC para o cálculo do trânsito de potências.

Na simulação pelo Método de Monte Carlo foi usado um sorteio cronológico para sortear as transições de estado do sistema.

O despacho foi feito proporcionalmente à potência disponível nos barramentos de produção. Ao analisar a saída de serviço das linhas quando havia linhas em sobrecarga, foi utilizado o método dos coeficientes de sensibilidade para tentar eliminar a sobrecarga através de um redespacho.

Quando surgem "ilhas" no sistema, foi considerado que era uma avaria global.

Foi definida uma lista de prioridades dos barramentos para o deslastre das cargas.

Foi apresentado o programa desenvolvido para os estudos de Fiabilidade do Sistema Composto, sendo apresentados os resultados para a rede teste para estudos de fiabilidade do IEEE.

Pelos resultados obtidos, verificou-se nomeadamente que a ordem de prioridade para deslastre dos barramentos influencia fortemente o cálculo dos índices de fiabilidade.

# 6. CONCLUSÃO

## 6.1 – Trabalho realizado

Neste trabalho, foram apresentados e discutidos alguns conceitos básicos de fiabilidade aplicados a Sistemas Eléctricos de Energia. Para exemplificar a aplicação destes conceitos, foram elaborados quatro programas, em linguagem C que permitem calcular a Fiabilidade do Sistema Produtor. Assim, os programas desenvolvidos permitem:

– Calcular o LOLE fazendo o truncamento das tabelas de probabilidades de ocorrência das capacidades fora de serviço;

– Calcular o LOLE fazendo o arredondamento das tabelas de probabilidades de ocorrência das capacidades fora de serviço;

– Calcular o LOLE pelo Método de Monte Carlo, usando uma visualização gráfica que permite ao utilizador verificar se houve ou não convergência para a dimensão da amostra utilizada;

– Calcular o LOLE pelo Método de Monte Carlo usando o coeficiente de variação como critério de convergência;

Foi ainda elaborado outro programa, também em linguagem C, para calcular a Fiabilidade do Sistema Composto. Este programa usou um sorteio cronológico através do qual foram sorteadas transições de estados do sistema. Foi usado o modelo DC para análise das violações dos fluxos nos ramos.

Os índices de fiabilidade obtidos foram os seguintes:

– Taxa média de avarias;

– Indisponibilidade anual;

– Tempo de reparação.

A linguagem C permite o uso de apontadores, o que foi fundamental na elaboração dos programas. Desta forma, consegue-se obter programas mais eficazes, principalmente quanto à complexidade do problema.

Os programas desenvolvidos foram testados com a rede teste de fiabilidades do IEEE. Foi feita posteriormente uma análise crítica dos programas desenvolvidos e das hipóteses assumidas.

## 6.2 – Trabalho futuro

Como complemento deste trabalho, expõe-se alguns assuntos que poderiam ser investigados futuramente e incluídos nos pressupostos do processo de simulação. Assim, pode-se considerar:

– A factorização ou bi-factorização de matrizes [7] ao calcular a matriz inversa de B. Desta forma, consegue-se melhorar o tempo computacional. Tal método poderá trazer vantagem significativa em sistemas de grandes dimensões;

– A análise dos fluxos nos ramos, mesmo quando há deslastres;

– A utilização do modelo AC para analisar os fluxos nos ramos, embora este processo seja mais penalizador em termos computacionais;

– A análise das "ilhas" como sistemas isolados;

– O uso de outras técnicas de redução da variância [6] e respectiva comparação de resultados;

– O mundo de recurso (alimentação alternativa), se existir;

– Avarias activas;

– Possibilidade de encravamento dos disjuntores;

– Acções de manutenção programada;

– O modelo de dois estados atmosféricos (normal e adverso). Neste modelo, pode-se ainda considerar as hipóteses de haver ou não reparação em tempo adverso e de a manutenção continuar ou não em tempo adverso [21];

– A influência das condições climatéricas das várias regiões onde o sistema está instalado;

– Estados de avarias parciais dos geradores, para além do estado de avaria total [6];

– A não inclusão do tempo em que o componente está fora de serviço no cálculo da taxa de avarias;

– A análise horária das cargas durante uma avaria do sistema. Este processo requer mais tempo computacional;

– A definição de uma política de deslastre das cargas que possa seguir outros critérios para além do de prioridade, tal como um critério económico que define se é preferível alimentar uma carga a custo elevado ou se é preferível deslastrá-la;

– O uso do método Simplex com o objectivo principal de minimizar os cortes [6]. Relativamente ao método utilizado no capítulo 5 (método dos coeficientes de sensibilidade), o método de Simplex consegue analisar um universo de oportunidades mais vasto, mas torna-se mais pesado em termos computacionais;

– A utilização de distribuições de probabilidade de um componente avariar diferentes da distribuição exponencial.

# Referências


1. ANDERS, George J., *Probability Concepts in Electric Power Systems*, Wiley, Nova Yorque, 1989.
2. ANDERS, George J.; ENDRENYI, J.; PEREIRA, Mário V. F.; PINTO, Leontina M. V. G.; OLIVEIRA, C. G.; CUNHA, S. H. F., *Techniques de Simulation Rapides par la Méthode de Monte Carlo pour les Etudes de Fiabilité des Réseaux*, 1990 CIGRÉ Meeting, paper 38-205, Paris, 1990.
3. BARBOSA, Fernando Pires Maciel, *Análise de Fiabilidade do Sistema Composto Produção/Transporte*, FEUP, Porto, 1987.
4. BILLINTON, Roy; ALLAN, Ronald N., *Reliability Evaluation of Power Systems*, Plenum Press, Nova Yorque, 1984.
5. BILLINTON, Roy; LI, Darui, *Composite System Reliability Evaluation Using Sequential Monte Carlo Simulation*, UPEC'98, págs. 146/149, 1998.
6. BILLINTON, Roy; LI, Wenyuan, *Reliability Assessment of Electric Power Systems Using Monte Carlo Methods*, Plenum Press, Nova Yorque, 1994.
7. BRAMELLER, A.; ALLAN, Ronald N.; HAMAM, Y. M., *Sparsity*, Pitman Publishing Limited, Londres, 1976.
8. CAMARGO, C. Celso de B., *Confiabilidade Aplicada a Sistemas de Potência Eléctrica*, LTC Editora, Rio de Janeiro, 1981.
9. DIALYNAS, Evangelos; KOSKOLOS, Nikolaos, *Comparison of Contingency Enumaration and Monte-Carlo Simulation Approaches Applied to the Reliability Evaluation of Composite Power Systems*, Diagnostic et Sûreté de Fonctionnement, vol. 5, n.º 1/1995, págs. 25/48, Atenas, 1995.
10. GAWLER, R. A.; MORSZTYN, K.; DILLON, T. S.; SMITH, D. C., *Application of the Monte Carlo Method to the Power System Reliability Evaluation*, Evaluation of Power System Reliability, paper n.º 3467, págs. 84/88, Institution of Engineers, 1976.
11. IEEE Committee Report, *IEEE Reliability Test System*, IEEE Transactions on Power Apparatus and Systems, vol. PAS-98, N.º 6, Novembro/Dezembro de 1979, págs. 2047/2054, 1979.
12. MATOS, Manuel, *Transparentes da Disciplina de Qualidade de Serviço de Sistemas Eléctricos de Energia*, FEUP, Porto, 1999.
13. MIRANDA, Vladimiro, *Principles of Monte Carlo Simulation*, FEUP, Porto, Maio de 1999.
14. PATTON, A. D.; BLACKSTONE, J. H.; BALU, Neal J., *A Monte Carlo Simulation Approach to the Reliability Modeling of Generating Systems Recognizing Operating Considerations*, IEEE Transactions on Power Apparatus and Systems, vol. 3, n.º 3/1988, págs. 1174/1180, 1987.
15. PEREIRA, Mário V. F.; BALU, Neal J., *Composite Generation/Transmission Reliability Evaluation*, IEEE Transactions on Power Apparatus and Systems, vol. 80, n.º 4/1992, págs. 464/491, 1991.

16.PEREIRA, Mário V. F.; MACEIRA, M. E. P.; OLIVEIRA, G. C.; PINTO, Leontina M. V. G., *Combining Analytical Models and Monte-Carlo Techniques in Probabilistic Power System Analysis*, IEEE Transactions on Power Apparatus and Systems.
17.PEREIRA, Mário V. F.; PINTO, Leontina M. V. G., *A New Computacional Tool for Composite Reliability Evaluation*, IEEE Transactions on Power Apparatus and Systems, vol. 7, n.º 1/1992, págs. 258/264.
18.PINTO, Leontina M. V. G.; PEREIRA, Mário V. F., *A Variance Reduction Technique to the Reliability Analysis of a Generation/Transmission System*, Rio de Janeiro.
19.*Reliability in Electric Power Systems*, homepage.htm.
20.SANTOS, José Eduardo Neves dos, *Transparentes da Disciplina de Complementos de Análise de Sistemas Eléctricos de Energia*, FEUP, Porto, 1998.
21.SILVA, José Luís Pereira da, *Transparentes da Disciplina de Qualidade de Serviço de Sistemas Eléctricos de Energia*, FEUP, Porto, 1999.
22.SILVA, José Luís Pereira da, *Fiabilidade de Redes de Distribuição de Energia Eléctrica Índices de Continuidade de Serviço e Rede Teste*, FEUP, Porto, Abril de 1993.
23.SILVA, José Luís Pereira da, *Transparentes da Disciplina de Complementos de Análise de Sistemas Eléctricos de Energia*, FEUP, Porto, 1998.
24.SOUSA, António Augusto Varejão Teixeira de, *Estudos de Fiabilidade Utilizando o Método de Simulação de Monte Carlo Integrando Informação Expressa sob a Forma de Números Imprecisos*, Dissertação de Mestrado, FEUP, Porto, Setembro de 1997.
25.SOUZA, Júlio César Stacchini de, *Análise da Confiabilidade Composta de Sistemas de Geração/Transmissão Incluindo Restrições de Segurança*, Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Agosto de 1991.

# Anexo A
# Codificação em C do programa 1 que calcula a Fiabilidade do Sistema Produtor

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

#define M 50

typedef struct
    {
    double pot,r;
    int n;
    } G;

typedef struct
    {
    double cfs,prob;
    } TAB;

int ler_dados(G [],double *,double *);
double capacidade_inst(G [],int);
void ler_prob_min(double *);
void continuar_abortar(TAB *);
TAB *construir_tab_prob(G [],int,int *,double);
int conjugar_tab(TAB *,int,TAB *,double);
void inserir(double,double,TAB *,int *,double);
int fcmp(TAB *,TAB *);
void calcular_lole(TAB *,int,double,double,double);
void cabecalho();
void soma_prob_desprezadas(TAB [],int);

main()
    {
    G g[M];
    double c1,c2,ci,prob_min;
    TAB *p;
    int ng,np;

    ng=ler_dados(g,&c1,&c2);
    ci=capacidade_inst(g,ng);
    while (ler_prob_min(&prob_min),prob_min>=0)
        {
        p=construir_tab_prob(g,ng,&np,prob_min);
        calcular_lole(p,np,c1,c2,ci);
        soma_prob_desprezadas(p,np);
```

```
        free(p);
        }
    }

int ler_dados(G g[],double *pc1,double *pc2)
    {
    FILE *f;
    char nom_f[81];
    int ng;
    double aux;

    clrscr();
    printf("Nome do ficheiro de texto que contém os dados? ");
    f=fopen(gets(nom_f),"rt");
    ng=0;
    while ((fscanf(f,"%lf",&aux),aux>0) && ng<M)
        {
        g[ng].pot=aux;
        fscanf(f,"%lf",&g[ng].r);
        fscanf(f,"%d",&g[ng].n);
        ng++;
        }
    if (aux>0)
        {
        printf("Mais de %d tipos de geradores\n",M);
        getch();
        exit(1);
        }
    fscanf(f,"%lf",pc1); fscanf(f,"%lf",pc2);
    return ng;
    }

double capacidade_inst(G g[],int ng)
    {
    int i;
    double s=0;

    for (i=0; i<ng; i++)
        s+=g[i].n*g[i].pot;
    return s;
    }

void ler_prob_min(double *pprob_min)
    {
    printf("\nProbabilidade mínima? (<0=sair) ");
    fflush(stdin); scanf("%lf",pprob_min);
    }

void continuar_abortar(TAB *p)
```

```
    {
    if (p==NULL)
        {
        printf("Impossível alocar memória\n");
        getch();
        exit(1);
        }
    }

TAB *construir_tab_prob(G g[],int ng,int *pnp,double prob_min)
    {
    TAB *p,q[2];
    int i,j,np;

    p=calloc(1,sizeof(TAB));
    continuar_abortar(p);
    p[0].cfs=0; p[0].prob=1;
    np=1;
    for (i=0; i<ng; i++)
        {
        q[0].cfs=0; q[0].prob=1-g[i].r;
        q[1].cfs=g[i].pot; q[1].prob=g[i].r;
        for (j=0; j<g[i].n; j++)
            {
            p=realloc(p,2*np*sizeof(TAB));
            continuar_abortar(p);
            np=conjugar_tab(p,np,q,prob_min);
            }
        }
    p=realloc(p,np*sizeof(TAB));
    continuar_abortar(p);
    *pnp=np;
    return p;
    }

int conjugar_tab(TAB *p,int np,TAB *q,double prob_min)
    {
    int i,j,k=2*np-1;
    double s;

    for (i=np-1; i>=0; i--)
        for(j=1; j>=0; j--)
            {
            p[k].cfs=p[i].cfs+q[j].cfs;
            p[k].prob=p[i].prob*q[j].prob;
            k--;
            }
    k=2*np;
    qsort(p,k,sizeof(TAB),fcmp);
```

```
    np=0; s=p[0].prob;
    for (i=1; i<k; i++)
        if (p[i].cfs==p[i-1].cfs)
            s+=p[i].prob;
        else
            {
            inserir(p[i-1].cfs,s,p,&np,prob_min);
            s=p[i].prob;
            }
    inserir(p[k-1].cfs,s,p,&np,prob_min);
    return np;
    }

void inserir(double cfs,double prob,TAB *p,int *pnp,double prob_min)
    {
    if (prob>=prob_min)
        {
        p[*pnp].cfs=cfs;
        p[*pnp].prob=prob;
        (*pnp)++;
        }
    }

int fcmp(TAB *p1,TAB *p2)
    {
    double aux=p1->cfs-p2->cfs;

    return -(aux<0)+(aux>0);
    }

void calcular_lole(TAB *p,int np,double c1,double c2,double ci)
    {
    int i,vis_tab;
    double cd,t,tp,s=0,aux=100.0/(c2-c1);
    char op;

    printf("Visualizar tabela? (s/n) "); op=getch();
    vis_tab=(op=='s') || (op=='S');
    if (vis_tab) cabecalho();
    for (i=0; i<np; i++)
        {
        cd=ci-p[i].cfs;
        if (cd>=c2)
            t=0;
        else
            if (cd<=c1)
                t=100;
            else
                t=aux*(c2-cd);
```

```
        tp=t*p[i].prob;
        s+=tp;
        if (vis_tab)
            {
            printf("%7.2lf    %8.6lf    %6.2lf    %10.6lf",
                    p[i].cfs,p[i].prob,t,tp);
            if (wherey()==25)
                {
                getch();
                cabecalho();
                }
            else
                putchar('\n');
            }
        }
    if (vis_tab && wherey()>21) getch();
    printf("\nLOLE = %0.6lf %%\n",s);
    }

void cabecalho()
    {
    clrscr();
    printf("   CFS       PROB       T (%)     T*PROB (%)\n");
    }

void soma_prob_desprezadas(TAB p[],int np)
    {
    int i;
    double s=1;

    for (i=0 ; i<np; i++)
        s-=p[i].prob;
    printf("Soma das probabilidades desprezadas = %0.2E\n",s);
    }
```

# Anexo B
# Codificação em C do programa 2 que calcula a Fiabilidade do Sistema Produtor

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#define M 50

typedef struct
    {
    double pot,r;
    int n;
    } G;

int ler_dados(G [],double *,double *);
double capacidade_inst(G [],int);
double pot_max(G [],int);
double ler_passo(double);
int int_acima(double);
int int_abaixo(double);
int const_vect_prob(double,double,int,double *,double *);
double ft(int);
int conjugar_vectores(double *,int,double *,int,int);
void calcular_lole(double *,int,double,double,double,double);
void cabecalho();

main()
    {
    G g[M];
    double c1,c2,passo,ci,pot_m,s,*p,*q;
    int ng,fp,fq,i;

    ng=ler_dados(g,&c1,&c2);
    ci=capacidade_inst(g,ng);
    pot_m=pot_max(g,ng);
    while ((passo=ler_passo(ci))>0)
        {
        p=calloc(int_acima(ci/passo)+1,sizeof(double));
        q=calloc(int_acima(pot_m/passo)+1,sizeof(double));
        if (p && q)
            {
            s=0;
            fp=const_vect_prob(g[0].pot/passo,g[0].r,g[0].n,p,&s);
            for (i=1; i<ng; i++)
```

```
                {
                fq=const_vect_prob(g[i].pot/passo,g[i].r,g[i].n,q,&s);
                fp=conjugar_vectores(p,fp,q,fq,int_acima(s));
                }
            calcular_lole(p,fp,c1/passo,c2/passo,ci/passo,passo);
            free(p); free(q);
            }
        else
            printf("Memória insuficiente. Escolha um passo maior.\n");
        }
    }

int ler_dados(G g[],double *pc1,double *pc2)
    {
    FILE *f;
    char nom_f[81];
    int ng;
    double aux;

    clrscr();
    printf("Nome do ficheiro de texto que contém os dados? ");
    f=fopen(gets(nom_f),"rt");
    ng=0;
    while ((fscanf(f,"%lf",&aux),aux>0) && ng<M)
        {
        g[ng].pot=aux;
        fscanf(f,"%lf",&g[ng].r);
        fscanf(f,"%d",&g[ng].n);
        ng++;
        }
    if (aux>0)
        {
        printf("Mais de %d tipos de geradores\n",M);
        getch();
        exit(1);
        }
    fscanf(f,"%lf",pc1); fscanf(f,"%lf",pc2);
    return ng;
    }

double capacidade_inst(G g[],int ng)
    {
    int i;
    double s=0;

    for (i=0; i<ng; i++)
        s+=g[i].n*g[i].pot;
    return s;
    }
```

```
double pot_max(G g[],int ng)
    {
    int i;
    double max=g[0].n*g[0].pot;

    for (i=1; i<ng; i++)
        if (g[i].n*g[i].pot>max)
            max=g[i].n*g[i].pot;
    return max;
    }

double ler_passo(double ci)
    {
    double passo;

    printf("\nCapacidade instalada=%0.0lf\n",ci);
    printf("Escolha o passo? (0=sair) ");
    fflush(stdin); scanf("%lf",&passo);
    return passo;
    }

int int_acima(double x)
    {
    return (int)ceil(x-1e-10);
    }

int int_abaixo(double x)
    {
    return (int)floor(x+1e-10);
    }

int const_vect_prob(double pot,double r,int n,double *v,double *ps)
    {
    int fv,i,i1,i2;
    double cfs,prob,pot_conjunto=pot*n;

    *ps+=pot_conjunto;
    fv=(int)ceil(pot_conjunto);
    for (i=0; i<=fv; i++)
        v[i]=0;
    for (i=0; i<=n; i++)
        {
        cfs=i*pot;
        prob=ft(n)/ft(i)/ft(n-i)*pow(r,i)*pow(1-r,n-i);
        i1=int_abaixo(cfs);
        i2=int_acima(cfs);
        if (i1==i2)
            v[i1]+=prob;
```

```
        else
            {
            v[i1]+=prob*(i2-cfs);
            v[i2]+=prob*(cfs-i1);
            }
        }
    return fv;
    }

double ft(int n)
    {
    double fact=1;

    for ( ; n>1; n--)
        fact*=n;
    return fact;
    }

int conjugar_vectores(double *p,int fp,double *q,int fq,int fp_max)
    {
    int i,k,i1,i2;
    double s=0;

    for (k=fp+fq; k>=0; k--)
        {
        if (k>fq) i1=k-fq; else i1=0;
        if (k<fp) i2=k; else i2=fp;
        for (i=i1; i<=i2; i++)
            s+=p[i]*q[k-i];
        if (k<=fp_max)
            {
            p[k]=s;
            s=0;
            }
        }
    return fp_max;
    }

void calcular_lole(double *p,int fp,double c1,double c2,double ci,double
passo)
    {
    int i,vis_tab;
    double cd,t,tp,s=0,aux=100.0/(c2-c1);
    char op;

    printf("Visualizar tabela? (s/n) "); op=getch();
    vis_tab=(op=='s') || (op=='S');
    if (vis_tab) cabecalho();
    for (i=0; i<=fp; i++)
```

```
        {
        cd=ci-i;
        if (cd>=c2)
            t=0;
        else
            if (cd<=c1)
                t=100;
            else
                t=aux*(c2-cd);
        tp=t*p[i];
        s+=tp;
        if (vis_tab)
            {
            printf("%7.2lf    %8.6lf    %6.2lf   %10.6lf",i*passo,p[i],t,tp);
            if (wherey()==25)
                {
                getch();
                cabecalho();
                }
            else
                putchar('\n');
            }
        }
    if (vis_tab && wherey()>21) getch();
    printf("\nLOLE=%0.6lf %%\n",s);
    }

void cabecalho()
    {
    clrscr();
    printf("   CFS      PROB      T (%)     T*PROB (%)\n");
    }
```

# Anexo C
# Codificação em C do programa 3 que calcula a Fiabilidade do Sistema Produtor

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <graphics.h>
#include <math.h>

#define M 50

typedef struct
    {
    float pot,r;
    int n;
    } G;

int ler_dados(G [],float *,float *);
void ler(char *, char *,void *,int);
void calcular_lole(G [],int,float,float,long,float,int);

main()
    {
    G g[M];
    int ng,gd,gm;
    float c1,c2,max;
    long n;
    char a[]="Dimensão da amostra? (0 - Ordenada max, <0 - Sair) ";

    clrscr();
    ng=ler_dados(g,&c1,&c2);
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"c:\\tc");
    ler("Ordenada máxima? ","%f",&max,gm);
    while (ler(a,"%ld",&n,gm),n>=0)
        if (n>0)
            calcular_lole(g,ng,c1,c2,n,max,gm);
        else
            ler("Ordenada máxima? ","%f",&max,gm);
    closegraph();
    }

int ler_dados(G g[],float *pc1,float *pc2)
    {
    FILE *f;
```

```
    char nom_f[81];
    int ng;
    float aux;

    clrscr();
    printf("Nome do ficheiro de texto que contém os dados? ");
    f=fopen(gets(nom_f),"rt");
    ng=0;
    while (ng<M && (fscanf(f,"%f",&aux),aux>0))
        {
        g[ng].pot=aux;
        fscanf(f,"%f",&g[ng].r);
        fscanf(f,"%d",&g[ng].n);
        ng++;
        }
    if (ng==M)
        {
        printf("Mais de %d tipos de geradores\n",M);
        getch();
        exit(1);
        }
    fscanf(f,"%f",pc1); fscanf(f,"%f",pc2);
    return ng;
    }

void ler(char *a, char *frm,void *p,int gm)
    {
    restorecrtmode();
    printf("%s",a);
    scanf(frm,p);
    setgraphmode(gm);
    }

void calcular_lole(G g[],int ng,float c1,float c2,long n,float max,int gm)
    {
    long i=0;
    int j,k;
    float cd1,cd2,st,s=0,aux=100.0/(c2-c1),kx=600.0/n,ky=400.0/max;
    char buf[11];

    randomize();
    setbkcolor(15);
    setcolor(1);
    outtextxy(0,15,"LOLE (%)");
    outtextxy(479,470,"NÚMERO DE SIMULAÇÕES");
    outtextxy(16,451,"0");
    outtextxy(607,451,"100%");
    rectangle(35,40,635,440);
    for (j=0; j<10; j++)
```

```
    outtextxy(0,36+j*40,gcvt((10-j)*max/10,3,buf));
do
    {
    i+=2;
    cd1=cd2=0;
    for (j=0; j<ng; j++)
        for (k=0; k<g[j].n; k++)
            {
            st=rand()/32768.0;
            if (st>g[j].r)
                cd1+=g[j].pot;
            if (1-st>g[j].r)
                cd2+=g[j].pot;
            }
    if (cd1<c2)
        if (cd1>c1)
            s+=aux*(c2-cd1);
        else s+=100;
    if (cd2<c2)
        if (cd2>c1)
            s+=aux*(c2-cd2);
        else s+=100;
    j=(int)(440.5-s/i*ky);
    if (j<40)
        j=40;
    putpixel((int)(35.5+i*kx),j,1);
    }
while (i<n);
getch();
restorecrtmode();
printf("Tamanho da amostra = %ld\n",i);
printf("LOLE = %0.6f\n",s/i);
getch();
setgraphmode(gm);
}
```

# Anexo D
# Codificação em C do programa 4 que calcula a Fiabilidade do Sistema Produtor

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define M 50

typedef struct
    {
    float pot,r;
    int n;
    } G;

int ler_dados(G [],float *,float *);
char menu(long,float,long);
void ler(char *,char *,void *);
void calcular_lole(G [],int,float,float,long,float,long);
float calcular_2lole(G *g,int,float,float);

main()
    {
    G g[M];
    int ng;
    float c1,c2,cv=0.001;
    long n=1000000,d=20000;
    char op;

    ng=ler_dados(g,&c1,&c2);
    while ((op=menu(n,cv,d))!=27)
        switch (op)
            {
            case '1': ler("Dimensão máxima da amostra?
","%ld",&n);break;
            case '2': ler("Coeficiente de variação? ","%f",&cv); break;
            case '3': ler("Duração da estabilidade? ","%ld",&d); break;
            case '4': calcular_lole(g,ng,c1,c2,n,cv,d); break;
            default : putchar('\7');
            }
    }

int ler_dados(G g[],float *pc1,float *pc2)
    {
```

```
    FILE *f;
    char nom_f[81];
    int ng;
    float aux;

    clrscr();
    printf("Nome do ficheiro de texto que contém os dados? ");
    f=fopen(gets(nom_f),"rt");
    ng=0;
    while (ng<M && (fscanf(f,"%f",&aux),aux>0))
        {
        g[ng].pot=aux;
        fscanf(f,"%f",&g[ng].r);
        fscanf(f,"%d",&g[ng].n);
        ng++;
        }
    if (ng==M)
        {
        printf("Mais de %d tipos de geradores\n",M);
        getch();
        exit(1);
        }
    fscanf(f,"%f",pc1); fscanf(f,"%f",pc2);
    return ng;
    }


char menu(long n,float cv,long d)
    {
    clrscr();
    gotoxy(17,8); printf("1 - Mudar dimensão máxima da amostra ( %ld
)",
                         n);
    gotoxy(17,10); printf("2 - Mudar coeficiente de variação ( %g )", cv);
    gotoxy(17,12); printf("3 - Mudar duração da estabilidade ( %ld )", d);
    gotoxy(17,14); printf("4 - Calcular lole por simulação\n");
    gotoxy(17,16); printf("Esc - Sair\n");
    gotoxy(17,18); printf("Opção? ");
    return getch();
    }

void ler(char *p1,char *p2,void *p3)
    {
    clrscr();
    printf(p1); scanf(p2,p3);
    }

void calcular_lole(G g[],int ng,float c1,float c2,long n,float cv,long d)
    {
```

```
    long i,j;
    float s1,lole,s2,var,med,desvio,cv2=cv*cv,cv2_med2;

    clrscr();
    randomize();
    s1=calcular_2lole(g,ng,c1,c2);
    i=2;
    med=lole=s1/2;
    s2=0;
    j=0;
    cv2_med2=cv2*med*med;
    do
        {
        s1+=calcular_2lole(g,ng,c1,c2);
        i+=2;
        lole=s1/i;
        desvio=lole-med;
        s2+=desvio*desvio;
        j++;
        var=s2/j;
        if (var>cv2_med2)
            {
            med=lole;
            s2=0;
            j=0;
            cv2_med2=cv2*med*med;
            }
        }
    while (i<n && j<d);
    printf("Número máximo de simulações = %ld\n",n);
    printf("Número de simulações = %ld\n",i);
    printf("Coeficiente de variação = %g\n",cv);
    printf("Duração da estabilidade = %ld\n",d);
    printf("LOLE (%%) = %0.6f\n",lole);
    getch();
    }

float calcular_2lole(G *g,int ng,float c1,float c2)
    {
    int i,j;
    float cd1=0,cd2=0,s=0,st,aux=100/(c2-c1);

    for (i=0; i<ng; i++)
        for (j=0; j<g[i].n; j++)
            {
            st=rand()/32768.0;
            if (st>g[i].r)
                cd1+=g[i].pot;
            if (1-st>g[i].r)
```

```
            cd2+=g[i].pot;
        }
if (cd1<c2)
    if (cd1>c1)
        s+=aux*(c2-cd1);
    else s+=100;
if (cd2<c2)
    if (cd2>c1)
        s+=aux*(c2-cd2);
    else s+=100;
return s;
}
```

# Anexo E
# Codificação em C do programa que calcula a Fiabilidade do Sistema Composto

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <mem.h>
#include <ctype.h>
#include <graphics.h>
#include <string.h>

#define MB 24
#define MB1 23
#define MG 32
#define MR 38
#define MC 17
#define MAV 11
#define MU 21
#define HORAS_ANO 8736
#define DZERO 1e-5

typedef struct
    {
    int bar;
    float max,min,tx[2];
    char e;
    } G;

typedef struct
    {
    int bar1,bar2;
    float max,x,tx[2];
    char e;
    } R;

typedef struct
    {
    int bar,prior;
    float pico;
    int av[MAV],u[MU],ava;
    long avtot;
    float ua,utot;
    char e;
    } C;
```

```
typedef struct
    {
    G *p1,*p2;
    float max,min,pg,pc,pi;
    } BAR;

typedef struct
    {
    int bar;
    float coef;
    } CS;

void ler_nb_sb(FILE *,int *,float *);
void abortar(char *);
int ler_dados_geradores(FILE *,G [],float);
int ler_dados_ramos(FILE *,R [],float);
int ler_dados_cargas(FILE *,C [],float);
void ler_diagrama_cargas(FILE *,float [],float []);
int simular(int,G [],int,R [],int,C [],int ,float [],float []);
int cmp_g_bar_min(G *,G *);
int cmp_c_prior(C *,C *);
int iniciar_bar_bg(G [],int,BAR [],int,int []);
float soma_picos(C c[],int);
void mat_bi(R [],int,float [][MB1],int);
void inverter_mat(float [][MB1],int);
void mat_a(R [],int,float [][MB1],int,float [][MB1]);
void mat_bii(float [][MB1],int,float []);
char sortear(G [],int,R [],int,float *,int *);
void atz_indisp(C [],int,float,char);
int atz_bar_bg(G *,BAR [],int [],int);
int cmp(float,float);
char viol_av(R [],int,C [],int,BAR [],int,int [],int,float [][MB1],float,
             float);
void deslastrar(C *);
void despacho(C [],int,float,int [],int,float,float,float,BAR [],int);
float calc_fl(int,float [][MB1],BAR [],int);
int fl_corr(float,float,int,float [][MB1],BAR [],int [],int);
int cmp_cs(CS *,CS *);
int diminuir(float,float,CS [],int,BAR []);
int aumentar(float,float,CS [],int,BAR []);
void manter_cargas(C [],int,int);
float fm(float,float [],float []);
int ha_ilhas(R [],int,int);
void deslastrar_cargas(C [],int,int);
void atz_mat_bi(float [][MB1],int,R *);
void mat_bi_inicial(float [],int,float [][MB1]);
void atz_anual(C [],int);
void reiniciar_mantendo_estado(C [],int);
```

```
void reiniciar_sem_deslastres(C [],int);
void calc_sair_resultados(C [],int,int);
int cmp_c_bar(C *,C *);
char menu();
void indices_fiab(FILE *,C [],int,int);
void probabilidades(FILE *,C [],int,int);
void graficos(C [],int,int);
FILE *abrir_fch();

main()
    {
    char nom_fich[81];
    FILE *fch;
    G g[MG];
    R r[MR];
    C c[MC];
    float sm[52],dia[7],sb;
    int nb,ng,nr,nc,anos;

    clrscr();
    printf("Nome do ficheiro de dados? ");
    fch=fopen(gets(nom_fich),"rt");
    ler_nb_sb(fch,&nb,&sb);
    ng=ler_dados_geradores(fch,g,sb);
    nr=ler_dados_ramos(fch,r,sb);
    nc=ler_dados_cargas(fch,c,sb);
    ler_diagrama_cargas(fch,sm,dia);
    close(fch);
    anos=simular(nb,g,ng,r,nr,c,nc,sm,dia);
    calc_sair_resultados(c,nc,anos);
    }

void ler_nb_sb(FILE *fch,int *pnb,float *psb)
    {
    fscanf(fch,"%d %f",pnb,psb);
    if (*pnb>MB)
        abortar("Demasiados barramentos");
    if (*psb<=0)
        abortar("Potência de base não positiva");
    }

void abortar(char *mensagem)
    {
    printf("%s\n",mensagem);
    getch();
    exit(1);
    }

int ler_dados_geradores(FILE *fch,G g[],float sb)
```

```
    {
    int n=0,aux1;
    float aux2,aux3,aux4,aux5;

    while ((fscanf(fch,"%d",&aux1),aux1>0) && n<MG)
        {
        fscanf(fch,"%f %f %f %f",&aux2,&aux3,&aux4,&aux5);
        g[n].bar=aux1-1;
        g[n].max=aux2/sb;
        g[n].min=aux3/100*g[n].max;
        g[n].tx[0]=HORAS_ANO/aux4;
        g[n].tx[1]=HORAS_ANO/aux5;
        g[n].e=0;
        n++;
        }
    if (aux1>0) abortar("Demasiados geradores");
    return n;
    }

int ler_dados_ramos(FILE *fch,R r[],float sb)
    {
    int n=0,aux1,aux2;
    float aux3,aux4,aux5,aux6;

    while ((fscanf(fch,"%d",&aux1),aux1>0) && n<MR)
        {
        fscanf(fch,"%d %f %f %f
%f",&aux2,&aux3,&aux4,&aux5,&aux6);
        r[n].bar1=aux1-1;
        r[n].bar2=aux2-1;
        r[n].max=aux3/sb;
        r[n].x=aux4;
        r[n].tx[0]=aux5;
        r[n].tx[1]=HORAS_ANO/aux6;
        r[n].e=0;
        n++;
        }
    if (aux1>0) abortar("Demasiados ramos");
    return n;
    }

int ler_dados_cargas(FILE *fch,C c[],float sb)
    {
    int i,n=0,aux1,aux2;
    float aux3;

    while ((fscanf(fch,"%d",&aux1),aux1>0) && n<MC)
        {
        memset(c+n,0,sizeof(C));
```

```
        fscanf(fch,"%d %f",&aux2,&aux3);
        c[n].bar=aux1-1;
        c[n].prior=aux2;
        c[n].pico=aux3/sb;
        n++;
        }
    if (aux1>0) abortar("Demasiadas cargas");
    return n;
    }

void ler_diagrama_cargas(FILE *fch,float sm[],float dia[])
    {
    int i;
    float aux;

    for (i=0; i<52; i++)
        {
        fscanf(fch,"%f",&aux);
        sm[i]=aux/100;
        }
    for (i=0; i<7; i++)
        {
        fscanf(fch,"%f",&aux);
        dia[i]=aux/100;
        }
    }

int simular(int nb,G g[],int ng,R r[],int nr,C c[],int nc,float sm[],
            float dia[])
    {
    float bi[MB][MB1],bii[MB1*MB1+MB1],a[MR][MB1],t,dt,spic;
    BAR bar[MB];
    int bg[MB],nbg,i,anos,ks,nb1=nb-1,avg=0,avr=0;
    char gr,e,es=0;

    /*
    Se não há deslastres es=0
    Se há deslastres mas não há ilhas es=1
    Se há ilhas es=2
    */
    printf("Tempo de simulação? (anos) "); scanf("%d",&anos);
    /* Ordenar os geradores por barramento e por potência mínima */
    qsort(g,ng,sizeof(G),cmp_g_bar_min);
    /* Ordenar as cargas por prioridades */
    qsort(c,nc,sizeof(C),cmp_c_prior);
    nbg=iniciar_bar_bg(g,ng,bar,nb,bg);
    spic=soma_picos(c,nc);
    memset(bi+nb1,0,MB1*sizeof(float));
    mat_bi(r,nr,bi,nb1);
```

```
mat_a(r,nr,bi,nb1,a);
mat_bii(bi,nb1,bii);
randomize();
gr=sortear(g,ng,r,nr,&dt,&ks);
for (i=1; i<=anos; i++)
    {
    t=0;
    while (t+dt<1)
        {
        atz_indisp(c,nc,dt,es); t+=dt;
        if (gr)
            {
            /* O componente ks é um ramo */
            e=r[ks].e=!r[ks].e; avr+=2*e-1;
            if (avr)
                if (ha_ilhas(r,nr,nb))
                    {
                    if (es<2) {es=2; deslastrar_cargas(c,0,nc-1);}
                    }
                else
                    {
                    if (es==2)
                        mat_bi(r,nr,bi,nb1);
                    else
                        atz_mat_bi(bi,nb1,r+ks);
                    mat_a(r,nr,bi,nb1,a);
                    es=viol_av(r,nr,c,nc,bar,nb,bg,nbg,a,spic,
                               fm(t,sm,dia));
                    }
            else
                {
                mat_bi_inicial(bii,nb1,bi); mat_a(r,nr,bi,nb1,a);
                if (avg)
                    es=viol_av(r,nr,c,nc,bar,nb,bg,nbg,a,spic,
                               fm(t,sm,dia));
                else
                    {es=0; manter_cargas(c,0,nc-1);}
                }
            }
        else
            {
            /* O componente ks é um gerador */
            e=g[ks].e=!g[ks].e; avg+=2*e-1;
            nbg=atz_bar_bg(g+ks,bar,bg,nbg);
            if (es<2)
                if (avg+avr)
                    es=viol_av(r,nr,c,nc,bar,nb,bg,nbg,a,spic,
                               fm(t,sm,dia));
                else
```

```
                    {es=0; manter_cargas(c,0,nc-1);}
                }
            gr=sortear(g,ng,r,nr,&dt,&ks);
            }
        atz_indisp(c,nc,1-t,es); dt-=1-t;
        atz_anual(c,nc);
        if (dt)
            reiniciar_mantendo_estado(c,nc);
        else
            reiniciar_sem_deslastres(c,nc);
        printf("%8d",i);
        }
    return anos;
    }

int cmp_g_bar_min(G *p1,G *p2)
    {
    int crit1=p1->bar-p2->bar;
    float crit2=p1->min-p2->min;

    return crit1? crit1 : -(crit2<0)+(crit2>0);
    }

int cmp_c_prior(C *p1,C *p2)
    {
    return p1->prior-p2->prior;
    }

int iniciar_bar_bg(G g[],int ng,BAR bar[],int nb,int bg[])
    {
    int i,k1,k2,nbg=0;
    G *p1,*p2;
    float s;

   memset(bar,0,nb*sizeof(BAR));
    /* Determinar p1 e p2 de cada barramento */
    k1=g[0].bar; bar[k1].p1=g;
    for (i=1; i<ng; i++)
        {
        k2=g[i].bar;
        if (k2!=k1)
            {
            bar[k1].p2=g+i-1; bar[k2].p1=g+i; k1=k2;
            }
        }
    bar[k2].p2=g+ng-1;
    /* Determinar max e min de cada barramento */
    for (i=0; i<nb; i++)
        {
```

```
        p1=bar[i].p1;
        if (p1)
            {
            bar[i].min=p1->min;
            s=0; p2=bar[i].p2;
            do
                {
                s+=p1->max; p1++;
                }
            while (p1<=p2);
            bar[i].max=s;
            bg[nbg]=i; nbg++;
            }
        }
    return nbg;
    }

float soma_picos(C c[],int nc)
    {
    int i;
    float s=0;

    for(i=0; i<nc; i++)
        s+=c[i].pico;
    return s;
    }

void mat_bi(R r[],int nr,float bi[][MB1],int nb1)
    {
    float y;
    int k1,k2,i,j;

    /* Construir a matriz b invertida (bi) */
    memset(bi,0,nb1*MB1*sizeof(float));
    for (i=0; i<nr; i++)
        if (!r[i].e)
            {
            y=1/r[i].x;
            k1=r[i].bar1;
            k2=r[i].bar2;
            j=0;
            if (k1<nb1)
                {
                bi[k1][k1]+=y; j++;
                }
            if (k2<nb1)
                {
                bi[k2][k2]+=y; j++;
                }
```

```
            if (j==2)
                bi[k1][k2]=bi[k2][k1]-=y;
            }
  inverter_mat(bi,nb1);
    }

void inverter_mat(float bi[][MB1],int nb1)
    {
    int i,j,k;

    for (i=0; i<nb1; i++)
        {
        bi[i][i]=1/bi[i][i];
        for (j=0; j<i; j++)
            {
            bi[j][i]*=bi[i][i];
            for (k=0; k<i; k++)
                bi[j][k]-=bi[j][i]*bi[i][k];
            for (k=i+1; k<nb1; k++)
                bi[j][k]-=bi[j][i]*bi[i][k];
            }
        for (j=i+1; j<nb1; j++)
            {
            bi[j][i]*=bi[i][i];
            for (k=0; k<i; k++)
                bi[j][k]-=bi[j][i]*bi[i][k];
            for (k=i+1; k<nb1; k++)
                bi[j][k]-=bi[j][i]*bi[i][k];
            }
        for (k=0; k<i; k++)
            bi[i][k]*=-bi[i][i];
        for (k=i+1; k<nb1; k++)
            bi[i][k]*=-bi[i][i];
        }
    }

void mat_a(R r[],int nr,float bi[][MB1],int nb1,float a[][MB1])
    {
    int i,j,k1,k2;
    float x;

    for (i=0; i<nr; i++)
        {
        k1=r[i].bar1; k2=r[i].bar2; x=r[i].x;
        for (j=0; j<nb1; j++)
            a[i][j]=(bi[k1][j]-bi[k2][j])/x;
        }
    }
```

```
void mat_bii(float bi[][MB1],int nb1,float bii[])
    {
    int i,j,k=0,nb2=nb1-1;

    for (k=0; k<nb1; k++)
        bii[k]=bi[k][k];
    for (i=0; i<nb2; i++)
        for (j=i+1; j<nb1; j++,k++)
            bii[k]=(bi[i][j]+bi[j][i])/2;
    }

char sortear(G g[],int ng,R r[],int nr,float *pdt,int *pks)
    {
    int i;
    float s1=0,s2=0,s3,s,sorteio;

    for(i=0; i<ng; i++)
        s1+=g[i].tx[g[i].e];
    for(i=0; i<nr; i++)
        s2+=r[i].tx[r[i].e];
    s3=s1+s2;
    *pdt=-1/s3*(float)log(1-(float)rand()/32768);
    sorteio=(float)rand()/32768*s3;
    if (sorteio<s1)
        {
        i=-1; s=0;
        do
            {
            i++; s+=g[i].tx[g[i].e];
            }
        while (s<sorteio);
        *pks=i;
        return 0;
        }
     else
        {
        i=-1; s=s1;
        do
            {
            i++; s+=r[i].tx[r[i].e];
            }
        while (s<sorteio);
        *pks=i;
        return 1;
        }
    }

 void atz_indisp(C c[],int nc,float dt,char es)
    {
```

```
    int i;

    if (es)
        for (i=0; i<nc; i++)
            c[i].ua+=dt*c[i].e;
    }

int atz_bar_bg(G *p,BAR bar[],int bg[],int nbg)
    {
    G *p1,*p2;
    int i,k=p->bar;
    float max;

    if (cmp(bar[k].max,0)>0)
        {
        max=0;
        for (p1=bar[k].p1, p2=bar[k].p2; p1<=p2; p1++)
            max+=p1->max*(1-p1->e);
        if (cmp(max,0)>0)
            {
            bar[k].max=max;
            p1=bar[k].p1;
            while (p1->e)
                p1++;
            bar[k].min=p1->min;
            }
        else
            {
            bar[k].max=bar[k].min=bar[k].pg=0;
            i=0;
            while (bg[i]!=k)
                i++;
            bg[i]=bg[nbg-1]; nbg--;
            }
        }
    else
        {
        bar[k].max=p->max; bar[k].min=p->min;
        bg[nbg]=k; nbg++;
        }
    return nbg;
    }

int cmp(float f1,float f2)
    {
    float aux=f1-f2;

    return -(aux<-DZERO)+(aux>DZERO);
    }
```

```
char viol_av(R r[],int nr,C c[],int nc,BAR bar[],int nb,int bg[],int nbg,
             float a[][MB1],float spic,float fm)
    {
    int i,k,fim,n=0,nb1=nb-1;
    char corr[MR],es;
    float smin=0,smax=0,ped=fm*spic,fl,dp,pd[MC],max;

    for (i=0; i<nbg; i++)
        {
        k=bg[i];
        smin+=bar[k].min; smax+=bar[k].max;
        }
    if (cmp(ped,smax)>0)
        {
        /* Potência pedida superior à potência disponível */
        do
            {
            pd[n]=fm*c[n].pico; ped-=pd[n];
            n++;
            }
        while (cmp(ped,smax)>0);
        /* Deslastrar barramentos podendo ainda recuperar alguns */
        dp=smax-ped;
        for (i=n-1; i>=0; i--)
            if (cmp(pd[i],dp)<=0)
                {c[i].e=0; dp-=pd[i];}
            else
                deslastrar(c+i);
        es=1;
        }
    else
        if (cmp(ped,smin)<0)
            /* Potência pedida inferior à soma dos mínimos dos
               barramentos */
            es=0;
        else
            {
            /* Potência pedida entre a soma dos mínimos e dos máximos
               dos barramentos */
            despacho(c,nc,fm,bg,nbg,smin,smax,ped,bar,nb);
            memset(corr,0,nr*sizeof(char));
            fim=i=0;
            do
                {
                if (!r[i].e)
                    {
                    fl=calc_fl(i,a,bar,nb1); max=r[i].max;
                    if (cmp(fabs(fl),max)>0)
```

```
                    if (!corr[i] && fl_corr(fl,max,i,a,bar,bg,nbg))
                        {fim=i; corr[i]=1;}
                    else
                        {deslastrar(c+n); n++;}
                }
            i=(i+1)%nr;
            }
        while (i!=fim && n==0);
        es=n>0;
        }
    manter_cargas(c,n,nc-1);
    return es;
    }

void deslastrar(C *p)
    {
    int aux=1-p->e;

    p->ava+=aux; p->avtot+=aux; p->e=1;
    }

void despacho(C c[],int nc,float fm,int bg[],int nbg,float smin,float smax,
                float ped,BAR bar[],int nb)
    {
    int i,k;
    float ds1=ped-smin,ds2=smax-ped,ds=smax-smin;
    G *p;

    /* A carga total está entre a soma das potências máximas e a soma das
    potências mínimas. Não houve deslastres */
    for (i=0; i<nbg; i++)
        {
        k=bg[i];
        bar[k].pg=(ds2*bar[k].min+ds1*bar[k].max)/ds;
        }
    for (i=0; i<nc; i++)
        {bar[c[i].bar].pc=fm*c[i].pico;}
    for (i=0; i<nb; i++)
        bar[i].pi=bar[i].pg-bar[i].pc;
    }

float calc_fl(int k,float a[][MB1],BAR bar[],int nb1)
    {
    int i;
    float s=0;

    for (i=0; i<nb1; i++)
        s+=a[k][i]*bar[i].pi;
    return s;
```

```
    }

int fl_corr(float fl,float max,int l,float a[][MB1],BAR bar[],
            int bg[],int nbg)
    {
    CS cs[MB];
    int i,k;

    for (i=0; i<nbg; i++)
        {
        k=bg[i]; cs[i].bar=k; cs[i].coef=a[l][k];
        }
    qsort(cs,nbg,sizeof(CS),cmp_cs);
    /* O fluxo está violado */
    if (fl>max)
        return diminuir(fl,max,cs,nbg,bar);
    else
        return aumentar(fl,-max,cs,nbg,bar);
    }

int cmp_cs(CS *p1,CS *p2)
    {
    float aux1=p1->coef,aux2=p2->coef;

    return -(aux1<aux2)+(aux1>aux2);
    }

int diminuir(float fl,float max,CS cs[],int nbg,BAR bar[])
    {
    int i1,i2,k1,k2;
    float dp1,dp2;

    i1=0; i2=nbg-1; k1=cs[i1].bar; k2=cs[i2].bar;
    while (i1<i2 && fl>max)
        {
        dp1=bar[k1].max-bar[k1].pg; dp2=bar[k2].pg-bar[k2].min;
        if (dp1<dp2)
            {
            bar[k1].pg=bar[k1].max; bar[k1].pi=bar[k1].pg-bar[k1].pc;
            bar[k2].pg-=dp1; bar[k2].pi=bar[k2].pg-bar[k2].pc;
            fl-=dp1*(cs[i2].coef-cs[i1].coef);
            i1++; k1=cs[i1].bar;
            }
        else
            {
            bar[k1].pg+=dp2; bar[k1].pi=bar[k1].pg-bar[k1].pc;
            bar[k2].pg=bar[k2].min; bar[k2].pi=bar[k2].pg-bar[k2].pc;
            fl-=dp2*(cs[i2].coef-cs[i1].coef);
            i2--; k2=cs[i2].bar;
```

```
            }
        }
    return cmp(fl,max)<=0;
    }

int aumentar(float fl,float max,CS cs[],int nbg,BAR bar[])
    {
    int i1,i2,k1,k2;
    float dp1,dp2;

    i1=0; i2=nbg-1; k1=cs[i1].bar; k2=cs[i2].bar;
    while (i1<i2 && fl<max)
        {
        dp1=bar[k1].pg-bar[k1].min; dp2=bar[k2].max-bar[k2].pg;
        if (dp1<dp2)
            {
            bar[k1].pg=bar[k1].min; bar[k1].pi=bar[k1].pg-bar[k1].pc;
            bar[k2].pg+=dp1; bar[k2].pi=bar[k2].pg-bar[k2].pc;
            fl+=dp1*(cs[i2].coef-cs[i1].coef);
            i1++; k1=cs[i1].bar;
            }
        else
            {
            bar[k1].pg-=dp2; bar[k1].pi=bar[k1].pg-bar[k1].pc;
            bar[k2].pg=bar[k2].max; bar[k2].pi=bar[k2].pg-bar[k2].pc;
            fl+=dp2*(cs[i2].coef-cs[i1].coef);
            i2--; k2=cs[i2].bar;
            }
        }
    return cmp(fl,max)>=0;
    }

void manter_cargas(C c[],int k1,int k2)
    {
    int i;

    for (i=k1; i<=k2; i++)
        c[i].e=0;
    }

float fm(float t,float sm[],float dia[])
    {
    int i,j;
    float aux;

    aux=t*52;
    i=(int)floor(aux);
    j=(int)floor((aux-i)*7);
    return sm[i]*dia[j];
```

```
    }

int ha_ilhas(R r[],int nr,int n)
    {
    int ilh[MB/2],*p_ilh[MB];
    int k1,k2,i,ni,nova_ilh;

    ni=0;
    nova_ilh=0;
    memset(p_ilh,0,n*sizeof(int *));
    for (i=0; i<nr; i++)
        if (r[i].e==0)
            {
            k1=r[i].bar1;
            k2=r[i].bar2;
            if (p_ilh[k1])
                if (p_ilh[k2])
                    {
                    if (*p_ilh[k1]!=*p_ilh[k2])
                        {*p_ilh[k1]=*p_ilh[k2]; ni--;}
                    }
                else
                    {
                    p_ilh[k2]=p_ilh[k1]; n--;
                    }
            else
                if (p_ilh[k2])
                    {
                    p_ilh[k1]=p_ilh[k2]; n--;
                    }
                else
                    {
                    ilh[nova_ilh]=nova_ilh;
                    p_ilh[k1]=p_ilh[k2]=ilh+nova_ilh;
                    nova_ilh++;
                    ni++; n-=2;
                    }
            }
    return ni!=1 || n>0;
    }

void deslastrar_cargas(C c[],int k1,int k2)
    {
    int i,aux;

    for (i=k1; i<=k2; i++)
        {
        aux=1-c[i].e; c[i].ava+=aux; c[i].avtot+=aux; c[i].e=1;
        }
```

```
    }

void atz_mat_bi(float bi[][MB1],int nb1,R *p)
    {
    int k1,k2,aux,i,j;
    float x1[MB1],x2[MB1],x,q;

    k1=p->bar1; k2=p->bar2; x=(1-2*p->e)*p->x;
    if (k1>k2)
        {
        aux=k1; k1=k2; k2=aux;
        }
    if (k2<nb1)
        {
        q=x+bi[k1][k1]+bi[k2][k2]-bi[k1][k2]-bi[k2][k1];
        for (i=0; i<nb1; i++)
            x1[i]=bi[i][k1]-bi[i][k2];
        for (i=0; i<nb1; i++)
            x2[i]=bi[k1][i]-bi[k2][i];
        }
    else
        {
        q=x+bi[k1][k1];
        for (i=0; i<nb1; i++)
            x1[i]=bi[i][k1];
        for (i=0; i<nb1; i++)
            x2[i]=bi[k1][i];
        }
    for (i=0; i<nb1; i++)
        for (j=0; j<nb1; j++)
            bi[i][j]-=x1[i]/q*x2[j];
    }

void mat_bi_inicial(float bii[],int nb1,float bi[][MB1])
    {
    int i,j,k=0,nb2=nb1-1;

  for (k=0; k<nb1; k++)
        bi[k][k]=bii[k];
    for (i=0; i<nb2; i++)
        for (j=i+1; j<nb1; j++,k++)
            bi[i][j]=bi[j][i]=bii[k];
    }

void atz_anual(C c[],int nc)
    {
    int i,j;

    for (i=0; i<nc; i++)
```

```
        {
        j=c[i].ava;
        if (j>=MAV) j=MAV-1;
            c[i].av[j]++;
        j=(int)floor(c[i].ua*HORAS_ANO/10);
        if (j>=MU) j=MU-1;
            c[i].u[j]++;
        c[i].utot+=c[i].ua;
        }
    }

void reiniciar_mantendo_estado(C c[],int nc)
    {
    int i;

    for (i=0; i<nc; i++)
        {
        c[i].ava=c[i].e;
        c[i].ua=0;
        }
    }

void reiniciar_sem_deslastres(C c[],int nc)
    {
    int i;

    for (i=0; i<nc; i++)
        {
        c[i].ava=0;
        c[i].ua=0;
        c[i].e=0;
        }
    }

void calc_sair_resultados(C c[],int nc,int anos)
    {
    FILE *sd;
    char op;

    qsort(c,nc,sizeof(C),cmp_c_bar);
    while ((op=menu())!=27)
        switch (toupper(op))
            {
            case 'I':
                indices_fiab(stdout,c,nc,anos);
                break;
            case 'P':
                probabilidades(stdout,c,nc,anos);
                break;
```

```
            case 'G':
                graficos(c,nc,anos);
                break;
            case 'F':
                sd=abrir_fch();
                if (sd)
                    {
                    indices_fiab(sd,c,nc,anos);
                    fprintf(sd,"\n");
                    probabilidades(sd,c,nc,anos);
                    fclose(sd);
                    gotoxy(31,13); printf("Resultados gravados ");
                    }
                else
                    {
                    gotoxy(23,13);
                    printf("Erro no nome do ficheiro ou desistiu ");
                    }
                getch();
                break;
            default : putchar('\7');
            }
    }

int cmp_c_bar(C *p1,C *p2)
    {
    return p1->bar-p2->bar;
    }

char menu()
    {
    char op;

    clrscr();
    gotoxy(28,8);  printf("I - Índices de fiabilidade");
    gotoxy(28,10); printf("P - Probabilidades");
    gotoxy(28,12); printf("G - Gráficos");
    gotoxy(28,14); printf("F - Gravar resultados num ficheiro");
    gotoxy(28,16); printf("Esc - Sair");
    gotoxy(28,18); printf("Opção? ");
    return getch();
    }

void indices_fiab(FILE *sd,C c[],int nc,int anos)
    {
    int i;
    float txav,u,tr;

    clrscr();
```

```
    fprintf(sd,"%54s\n","BAR   TX AV   INDISP   TEMP R");
    for (i=0; i<nc; i++)
        {
        txav=(float)c[i].avtot/anos;
        u=c[i].utot*HORAS_ANO/anos;
        tr=u/txav;
        fprintf(sd,"%28d%8.2f%9.2f%9.2f",c[i].bar+1,txav,u,tr);
        if (wherey()==25)
            {
            getch(); clrscr();
            }
        else
            fprintf(sd,"\n");
        }
    if (sd==stdout && wherey()<25) getch();
    }

void probabilidades(FILE *sd,C c[],int nc,int anos)
    {
    int i,j,m=max(MAV,MU);

    clrscr();
    for (i=0; i<nc; i++)
        {
        fprintf(sd,"%43s %d\n","BARRAMENTO",c[i].bar+1);
        fprintf(sd,"%57s\n","AV/ANO  PROB(%)  INDISP  PROB(%)");
        for (j=0; j<m; j++)
            {
            if (j<MAV)
                {
                if (j==MAV-1)
                    fprintf(sd,"%28s","Resto");
                else
                    fprintf(sd,"%28d",j);
                fprintf(sd,"%10.1f",(float)c[i].av[j]/anos*100);
                }
            else
                fprintf(sd,"%38s","");
            if (j<MU)
                {
                if (j==MU-1)
                    fprintf(sd,"%9s","Resto");
                else
                    fprintf(sd,"%9d",10*j+5);
                fprintf(sd,"%10.1f",(float)c[i].u[j]/anos*100);
                }
            fprintf(sd,"\n");
            }
        if (sd==stdout) {getch(); clrscr();} else fprintf(sd,"\n");
```

```
    }
  }
void graficos(C c[],int nc,int anos)
  {
  int gd,gm,i,j,x1,x2,dxu[]={10,6,2},dxav[]={9,1};
  float amp;
  char str[4],strbar[16];

  clrscr();
  detectgraph(&gd,&gm);
  initgraph(&gd,&gm,"c:\\tc");
  setbkcolor(15);
  setcolor(1);
  setfillstyle(SOLID_FILL,1);
  for (i=0; i<nc; i++)
      {
      cleardevice();
      line(24,208,620,208); line(24,208,24,0); line(24,8,620,8);
      outtextxy(254,224,"Avarias/Ano");
      outtextxy(12,204,"0"); outtextxy(12,4,"1");
      line(24,455,620,455); line(24,455,24,247); line(24,255,620,255);
      outtextxy(254,471,"Indisponibilidade");
      outtextxy(12,451,"0"); outtextxy(12,251,"1");
      settextstyle(DEFAULT_FONT,VERT_DIR,1);
      outtextxy(8,52,"Probabilidade");
      outtextxy(8,299,"Probabilidade");
      strcpy(strbar,"BARRAMENTO "); itoa(c[i].bar+1,str,10);
      strcat(strbar,str);
      outtextxy(636,180,strbar);
      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
      for (j=0; j<11; j++)
          {
          x1=50+j*52; x2=76+j*52;
          amp=(float)c[i].av[j]/anos;
          bar(x1,208-(int)(200*amp+0.5),x2,208);
          if (j<10) itoa(j,str,10); else strcpy(str,"Res");
          outtextxy(x1+dxav[j>9],212,str);
          }
      for (j=0; j<21; j++)
          {
          x1=24+j*28; x2=52+j*28;
          amp=(float)c[i].u[j]/anos;
          bar(x1,455-(int)(200*amp+0.5),x2,455);
          if (j<20) itoa(5+10*j,str,10); else strcpy(str,"Res");
          outtextxy(x1+dxu[(j>0)+(j>9)],459,str);
          }
      getch();
      }
```

```
    closegraph();
    }

FILE *abrir_fch()
    {
    char nom_fch[81];

    clrscr();
    fflush(stdin);
    printf("Nome do ficheiro? (enter sozinho=desistir) "); gets(nom_fch);
    return fopen(nom_fch,"wt");
    }
```