UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE PESQUISAS HIDRÁULICAS

INSTRUMENTALIZAÇÃO DO PROPAGAR MOO COM FERRAMENTAS DE PLANEJAMENTO DO USO DA ÁGUA E DE ANÁLISE DA SIMULAÇÃO ATRAVÉS DA UTILIZAÇÃO DA LINGUAGEM PASCAL SCRIPT.

FLÁVIO HADLER TRÖGER

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental da Universidade Federal do Rio Grande do Sul como requisito parcial para a obtenção do título de Mestre em Engenharia.

Orientador: Prof. Dr. Antônio Eduardo Leão Lanna
Co-orientador: Prof. Dr. João Soares Viegas Filho

Banca Examinadora:

Prof. Dr. Alexandre Beluco	(IPH/UFRGS)
Prof. Dr. Amauri de Almeida Machado	(IFM/UFPel)
Prof. Dr. André Lopes da Silveira	(IPH/UFRGS)

Porto Alegre, Abril de 2002.

APRESENTAÇÃO

Este trabalho foi desenvolvido no Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental do Instituto de Pesquisas Hidráulicas da Universidade Federal do Rio Grande do Sul, sob a orientação do Prof. Doutor Antônio Eduardo Leão Lanna da Universidade Federal do Rio Grande do Sul e co-orientado pelo Prof. Doutor João Soares Viegas Filho da Universidade Federal de Pelotas.

AGRADECIMENTOS

Agradeço aos professores do Instituto de Pesquisas Hidráulicas da UFRGS por sua dedicação e profissionalismo, tornando essa instituição uma referência nacional no estudo dos recursos hídricos.

Faço um agradecimento à CAPES pela bolsa de mestrado concedida, o que possibilitou a realização do curso de pós-graduação.

Agradeço ao Prof. Doutor Antônio Eduardo Leão Lanna pela orientação recebida durante os trabalhos e pelo aprendizado que disso resultou.

Faço um agradecimento especial ao professor, colega e amigo, Dr. João Soares Viegas Filho, co-orientador deste trabalho, por sua valiosa orientação, apoio, paciência e incentivo durante essa jornada.

Agradeço aos professores Vitor Tavares e Rita Fraga Damé, da Universidade Federal de Pelotas, pela motivação e amizade.

Agradeço à secretária da Pós-graduação, Nadir Solari, por sua exemplar dedicação ao Programa de Pós-Graduação.

Aos colegas André, Beatriz, Käthe, Ana Paula, Jorge Pilar, Hansen, Walter Collischon, Alex, Jaildo, Cleuda, Luciana, Roberto, Brusa, Carlos, Walter Vianna, Mário e Marcus pela amizade e pelo companheirismo durante o curso.

Ao Bacharel em Informática Adriano Rochedo Conceição pelo valioso apoio técnico na área de programação.

Agradeço aos meus irmãos, Henrique (e Jaqueline), Anelise (e Ubirajara) e Marta, por sua eterna amizade, carinho e incentivo.

Aos meus pais, Carl e Lilia, por ter recebido sempre, muito mais do que lições de amor, carinho, amizade, compreensão, moral, honestidade e compromisso: OBRIGADO PELO SEU EXEMPLO.

Agradeço a Lauren por todo o amor, compreensão, apoio e carinho que recebo, estando sempre ao meu lado como uma verdadeira companheira.

Por fim, agradeço a Deus por participar e crescer na mais importante de todas as escolas: “A escola da vida”, onde eventualmente somos professores, porém, eternamente alunos.

RESUMO

A simulação é uma das ferramentas mais utilizadas para a aplicação da análise sistêmica nos mais diversos estudos. Ao longo do tempo, vários modelos foram desenvolvidos para representar sistemas de recursos hídricos, utilizando a simulação.

Dentre esses modelos, está o Propagar MOO, que simula a propagação de vazões em uma bacia hidrográfica, submetida à decisões operacionais de suprimento de demandas e de operação de reservatórios, introduzidas pelo usuário através de rotinas escritas na linguagem de programação Pascal Script. A utilização eficiente dessas rotinas permite ao usuário ampliar a capacidade e flexibilidade do modelo na representação de um sistema hídrico.

Com o objetivo de contribuir na ampliação da flexibilidade do modelo Propagar MOO e de sua aplicabilidade à modelagem de sistemas de recursos hídricos em geral, bem como facilitar o estudo da linguagem de programação Pascal Script e motivar os profissionais da área no desenvolvimento de novas rotinas aplicadas ao modelo, foram implementadas, através do presente trabalho, rotinas genéricas contendo estratégias de planejamento do uso da água e de operação de reservatórios, bem como ferramentas para analisar seus resultados.

Para ampliar essa contribuição, foi aprimorada a possibilidade de simulação da geração de energia hidrelétrica em pontos de uma rede hidrográfica, com a criação de novas ferramentas para esse fim, na estrutura interna do modelo.

Por fim, para que o próprio usuário pudesse construir ferramentas para auxiliar na verificação dos resultados obtidos nas simulações, esse estudo apresenta a implementação de rotinas de uso geral para servir como exemplos de ferramentas de análise de dados.

ABSTRACT

Simulation is one of the most used tools in the application of systemic analysis in a broad range of studies. Along the time, several models were developed to represent water resources systems using simulation.

Among these models, it is Propagar MOO, which simulates the flow propagation in a river basin under operational decisions of demand fulfillment and of reservoirs operation, introduced by the user through routines written in Pascal Script programming language. The efficient use of these routines allows the user to expand the capacity and flexibility of the model to represent a water system.

In the present work, were implemented generic routines containing strategies of water use planning and reservoirs operation, as well as tools to analyze their results. The main objective is to improve the flexibility of the Propagar MOO model and its suitability for the modeling of water resources systems in general. Another objective is to facilitate the study of the Pascal Script programming language and motivate the professionals in the water resources area to develop new routines to apply in the model.

Moreover, it was improved the possibility of simulation of hydroelectric power generation at points of a water system, through the creation of new specific tools in the internal framework of the model.

Finally, aiming to allow the user to build tools to aid in the verification of the results obtained in simulations, this study presents the implementing of routines of general use to act as examples of tools for data analysis.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

CAPÍTULO 1 - INTRODUÇÃO.

1.1 MOTIVAÇÃO DO ESTUDO.

Devido ao aumento das demandas e da diversidade de usos dos recursos hídricos disponíveis, torna-se cada vez mais importante conduzir de forma sistêmica quaisquer estudos na área de recursos hídricos, assim como a gestão de qualquer aproveitamento hídrico. Uma das metodologias mais usadas para abordar de forma sistêmica um determinado estudo é a simulação. Através dela, foram desenvolvidos vários modelos para representar sistemas de recursos hídricos, como por exemplo, o Propagar MOO.

O Propagar MOO baseia-se em uma estrutura de Rede Hidrográfica, constituída por Pontos Característicos (PCs) ligados entre si por Trechos de Água. O modelo simula a propagação de vazões em uma bacia hidrográfica, submetida à decisões operacionais de suprimento de demandas hídricas e de operação de reservatórios. Estas são introduzidas pelo usuário, através de “scripts” (rotinas simples, escritas em uma linguagem de programação apropriada e que servem para a programação de pequenos algoritmos em programas complexos) para adequar o modelo às suas necessidades.

A utilização eficiente dessas rotinas permite ao usuário ampliar a capacidade e flexibilidade do modelo na representação de um sistema hídrico qualquer. Entretanto, o estudo de uma linguagem de programação demanda tempo e esforço que nem sempre os usuários dispõem e, além disso, o uso de exemplos já construídos é um fator facilitador para o estudo e desenvolvimento de rotinas em uma linguagem de programação.

Assim, verificou-se a necessidade de implementar rotinas genéricas contendo estratégias de planejamento do uso da água e de operação de reservatórios, com possibilidade de aplicação em diferentes estudos, bem como ferramentas para analisar seus resultados.

Verificou-se, também, a importância de aprimorar a possibilidade de simulação da geração de energia hidrelétrica em uma rede hidrográfica construída no modelo Propagar, devido à importância adquirida pelo tema no atual cenário brasileiro. Para isso, seria necessário desenvolver novas ferramentas, no corpo do modelo, para simular e quantificar a geração de energia hidrelétrica em pontos de uma rede hidrográfica.

Por fim, para complementar a evolução que o modelo teria com a implementação das propostas apresentadas acima, seria fundamental que o próprio usuário pudesse construir ferramentas para auxiliar na análise dos resultados obtidos nas simulações, bem como utilizar com eficiência os “Scripts Gerais” como ferramenta de análise de dados.

1.2 OBJETIVOS.

Os propósitos acima explicitados motivaram os seguintes objetivos formais a serem alcançados pelo presente trabalho:

1.2.1 Objetivos gerais:

- Desenvolver ferramentas de planejamento e gestão do uso da água, que permitam ao usuário interagir com o modelo de simulação Propagar MOO;
- Possibilitar aos usuários do modelo Propagar MOO, a simulação da geração de energia hidrelétrica, bem como avaliar seus resultados;
- Orientar os usuários do modelo no desenvolvimento de novas ferramentas.

1.2.2 Objetivos Específicos:

- Implementar ferramentas genéricas de planejamento do uso da água e operação de reservatórios no modelo Propagar MOO, utilizando a linguagem Pascal Script;
- Implementar no modelo Propagar MOO um método padrão para simular e quantificar a geração de energia hidrelétrica em uma rede hidrográfica e, a partir deste, propor funções e procedimentos em Pascal Script específicos para este fim, permitindo ao usuário a implementação e utilização de métodos alternativos;
- Implementar ferramentas de auxílio à análise de resultados da simulação no modelo Propagar MOO;
- Orientar os usuários do modelo Propagar MOO no processo de desenvolvimento de ferramentas aplicadas, utilizando as chamadas de rotinas em Pascal Script que o modelo possui.

1.3 ORGANIZAÇÃO DO TEXTO.

O presente trabalho apresenta os assuntos que fundamentam, desenvolvem e exemplificam o tema proposto, além de conduzir de forma didática a abordagem dos tópicos que o compõe. Assim sendo, este item tem o propósito de expor ao leitor a forma como o texto foi organizado, procurando facilitar o seu entendimento.

O presente capítulo, denominado “Capítulo 1 – Introdução”, serve de introdução ao trabalho e inicia pela motivação do estudo, mostrando sua pertinência e justificativa, sendo seguido, pelos objetivos estabelecidos para o trabalho. No final, indica ao leitor a estrutura do texto e caracteriza a contribuição que o trabalho traz.

O “Capítulo 2 – Revisão Bibliográfica” apresenta, inicialmente, aspectos gerais sobre a gestão e o planejamento de recursos hídricos, bem como a necessidade de uma abordagem sistêmica nos projetos e estudos envolvendo os recursos hídricos e as principais metodologias utilizadas. Ainda são revistos, alguns métodos de simulação aplicados a sistemas de recursos hídricos e conceitos de estratégias operacionais para o planejamento do uso da água. Também é abordada a concepção básica do modelo Propagar MOO (modelo construído segundo os princípios da Modelagem Orientada a Objetos), sua dinâmica e suas principais ferramentas de análise de resultados. Por fim, o capítulo apresenta a linguagem Pascal Script, utilizada na construção de rotinas aplicadas ao modelo Propagar MOO.

O “Capítulo 3 – Metodologia” apresenta, sob o ponto de vista conceitual, os métodos que foram desenvolvidos e implementados na linguagem de programação Pascal Script, com objetivo de instrumentalizar o modelo Propagar com ferramentas genéricas de planejamento do uso da água e de operação de reservatório, bem como os procedimentos e funções cuja criação se fez necessária para a implementação de tais ferramentas. Este capítulo também apresenta novas ferramentas para a simulação da geração de energia hidrelétrica em PCs de uma rede hidrográfica, além de demonstrar algumas maneiras de como o próprio usuário pode construir novas ferramentas para análise dos resultados de simulações.

No “Capítulo 4 – Aplicação” apresenta-se a aplicação das ferramentas de planejamento do uso da água e operação de reservatórios, de simulação da geração de energia e de análise da simulação, desenvolvidas no presente trabalho, bem como os resultados alcançados, com o propósito de lhes conferir validação.

Para realizar simulações aplicando as ferramentas desenvolvidas foi necessário representar um sistema hídrico no modelo Propagar MOO. Para tal, foi escolhida a bacia hidrográfica do rio Paracatu, afluente do rio São Francisco, devido à disponibilidade de dados hidrológicos e de informações sobre a bacia.

O “Capítulo 5 – Conclusões e Recomendações”, procura fazer uma análise crítica dos resultados alcançados pelo trabalho como um todo, buscando compatibilizá-los com os objetivos traçados, extraindo conclusões e recomendações que possam servir de orientação para outros trabalhos que, eventualmente, venham dar continuidade a este.

1.4 CONTRIBUIÇÃO DO ESTUDO.

O estudo desenvolvido vem contribuir na ampliação da flexibilidade do modelo Propagar MOO e de sua aplicabilidade à modelagem de sistemas de recursos hídricos em geral, bem como no desenvolvimento de ferramentas aplicadas ao modelo, na medida em que:

1. Apresenta ferramentas genéricas de planejamento do uso da água e operação de reservatórios, aplicadas ao modelo Propagar MOO;

2. Apresenta novos métodos e funções auxiliares implementados para a linguagem Pascal Script e disponibilizados no Editor Pascal Script do Propagar MOO;

3. Acrescenta à estrutura do modelo Propagar MOO uma metodologia padrão que permite a simulação e avaliação da geração de energia hidrelétrica em qualquer ponto característico de uma rede hidrográfica;

4. Apresenta ferramentas que permitem ao usuário utilizar formas alternativas para simular e avaliar a geração de energia hidrelétrica em pontos de uma rede hidrográfica;

5. Apresenta ferramentas para auxiliar na análise dos resultados de processos de simulação utilizando o Propagar MOO;

6. Orienta os usuários do modelo Propagar MOO no processo de desenvolvimento de novas ferramentas aplicadas ao modelo, utilizando a linguagem de programação Pascal Script.

CAPÍTULO 2 - REVISÃO BIBLIOGRÁFICA

2.1 ASPECTOS GERAIS SOBRE A GESTÃO E PLANEJAMENTO DE USO DA ÁGUA

2.1.1 Considerações gerais

Pode-se criar inúmeras categorias para classificar os diferentes usos da água no mundo atual, como por exemplo, consumo humano, abastecimento de animais, irrigação, indústria, navegação e geração de energia. Pode-se, ainda, incluir nessa lista as atividades recreativas que a água propicia ou simplesmente o lazer contemplativo, pois ela é um elemento fundamental da paisagem e do meio ambiente.

Também é comum entre os autores a classificação dos usos quanto à natureza da utilização, em consuntivos e não-consuntivos. No primeiro caso, a água é derivada do seu curso natural diminuindo sua disponibilidade quantitativa (podendo ocorrer alteração na sua qualidade), como acontece com a irrigação ou o abastecimento urbano. No segundo caso, a água pode atender à demanda sem ser retirada da fonte de suprimento, ou, ainda, ser retirada do manancial e, após o atendimento das demandas, ser totalmente devolvida. Como exemplo desse caso pode-se citar a geração de energia.

Lanna (1997a) ainda descreve uma terceira classe quanto à natureza da utilização, chamada local, referindo-se aos usos que aproveitam a disponibilidade de água em sua fonte sem qualquer modificação relevante, temporal ou espacial, de disponibilidade quantitativa. Os usos que se enquadrariam nessa classe normalmente são classificados como não-consuntivos por outros autores.

Com a ampliação considerável da diversidade de usos da água, as disponibilidades que eram inicialmente utilizadas para o suprimento de um único propósito passaram a ser objetos de interesses diversos, o que é favorecido pelo chamado efeito multiplicador dos projetos, descrito por Lanna (1997a), onde um determinado projeto inicial, como de irrigação, por exemplo, provoca outras demandas ao tornar mais atrativa economicamente a região onde foi implantado.

O problema complica-se mais ainda pela existência de outros conflitos operacionais, como os que se apresentam no caso de suprimento hídrico quando se deve

decidir que parte da reserva de água será usada no presente e que parte será preservada para seu uso futuro (Lanna, 1982).

A integração harmônica dos diversos usos visando adequar o seu padrão temporal de utilização, tem como alternativa os conflitos entre usuários, pois como afirma Lanna (1997a), o uso múltiplo dos recursos hídricos não é uma opção que faz o planejador, mas uma realidade que ele enfrenta com o desenvolvimento econômico. Na verdade, o que se busca é uma maneira justa de distribuir a água entre os diversos usuários nos períodos críticos, evitando os possíveis conflitos.

Existem vantagens no uso múltiplo e integrado dos recursos hídricos, como a de que a capacidade final do sistema pode não ser, necessariamente, igual à soma das capacidades individuais dos sistemas que atenderiam a um único uso, citada por Lanna (1997c), sendo possível que o sistema integrado possa suprir mais demandas. Outra vantagem é a possibilidade de serem obtidas economias de escala captadas na implantação do sistema, quando os custos de investimento, operação e manutenção por unidade da dimensão do projeto diminuem com a dimensão total. As desvantagens apresentadas são de caráter gerencial, como a exigência do estabelecimento de regras operacionais geralmente complexas para manter a harmonia entre os usos, havendo também a necessidade de centralização das decisões.

Nessa mesma linha, Barth (1987) afirma que os usos para múltiplos fins são simultâneos e com efeitos cumulativos. As alterações qualitativas e quantitativas ocorrem de forma integrada, porém a ciência humana, pelas suas próprias limitações, tende a segmentar as partes de um determinado sistema para conseguir estudá-lo.

Na verdade, os diversos usos da água estão, direta ou indiretamente, voltados ao atendimento das necessidades humanas, porém, muitos desses usos não são totalmente compatíveis entre si e alguns até podem ser mutuamente excludentes, podendo gerar situações de conflito.

Diante disso, surge a necessidade de se adotar normas e leis visando assegurar uma divisão justa dos recursos hídricos disponíveis, bem como solucionar possíveis conflitos originados pelas diferenças de interesse entre os usuários. O estabelecimento de diretrizes e normas para gerenciar a utilização dos recursos hídricos, bem como a sua implantação e execução, faz parte do que se convencionou chamar de Gestão dos Recursos Hídricos.

2.1.2 O Planejamento e a Gestão de Recursos Hídricos no contexto atual

A diversidade de usos tornou necessária a evolução dos sistemas de recursos hídricos de forma que, apoiados na pesquisa e nos avanços científicos e tecnológicos, melhor refletissem essa nova realidade.

Os planos futuros exigem algo substancialmente melhor do que uma simples proposta de projeto que vise tão somente uma relação benefício-custo favorável. A grande pressão pela terra e água como fatores de produção muito valorizados, o desenvolvimento da responsabilidade ambiental e as múltiplas possibilidades de uso da água fazem com que, no planejamento de novos projetos de sistemas de recursos hídricos, sejam desenvolvidas e utilizadas ferramentas que visem a otimização da alocação dos recursos disponíveis, sugerindo formas mais eficientes de atingir os mesmos objetivos.

Sendo o uso múltiplo dos recursos hídricos uma realidade e sua integração uma necessidade, cabe aos profissionais que trabalham na área de recursos hídricos dedicarem mais atenção e recursos à etapa de planejamento. Esse planejamento visa a avaliação das demandas e das disponibilidades dos recursos hídricos, bem como, a alocação da água entre os diversos usos, de forma a obter os máximos benefícios sociais e econômicos.

De acordo com Porto (1997), a essência do problema de planejamento em recursos hídricos está em como tomar decisões acertadas a respeito de um campo que se caracteriza pela complexidade, incertezas de diversas naturezas, existência de conflitos, investimentos de porte elevado, necessidade de planejamento a longo prazo, dinamismo ao longo da vida útil dos sistemas, repercussões econômicas, sociais e ambientais significativas e pela participação de grupos heterogêneos no processo decisório.

Na realidade, não existe uma metodologia universal a aplicar no planejamento de recursos hídricos, pois a alocação de água entre usos múltiplos é um problema bastante complexo, cuja solução deveria ser procurada aplicando técnicas que tratassem os problemas dentro de uma abordagem sistêmica.

Uma afirmação desse fato pode ser obtida no trabalho de Yeh (1985) onde o autor faz uma revisão aprofundada dos modelos de gerenciamento e operação de reservatórios e

uma de suas observações foi que, cada problema de planejamento e operação de reservatórios é único, não existindo um algoritmo geral ou universal para a sua resolução.

Devido à característica dinâmica dos aproveitamentos hídricos, alguns autores como Grull (1981) fazem interessantes sugestões, como a de que, no decorrer do tempo, após a implantação dos projetos, as hipóteses hidrológicas adotadas devem ser verificadas face à existência de séries observadas mais longas, bem como a evolução das técnicas de tratamento de dados. Mas a dinâmica dos aproveitamentos hídricos não se deve unicamente à evolução das séries hidrológicas ou a novas ferramentas de análise, mas, também, ao surgimento ou alterações nas demandas inicialmente previstas, seja durante os estudos de um determinado projeto, seja em projetos já implantados.

No caso de projetos já implantados, cabe ao seu grupo gestor manter uma política de planejamento constante que permita fazer previsões, testar hipóteses e avaliar as suas conseqüências, pois não seria prudente para a instituição responsável pela gestão de um determinado aproveitamento hídrico ou de uma bacia hidrográfica manter um comportamento passivo, esperando o surgimento real de novas demandas para, só então, estudar a possibilidade de atendê-las. Esse tipo de comportamento torna a instituição vulnerável, podendo comprometer a eficiência do sistema e inviabilizar soluções rápidas e práticas. O ideal é tentar antecipar-se às necessidades.

Muitos são os exemplos que demonstram a indissociabilidade entre o processo de planejamento de um determinado aproveitamento hídrico e a sua gestão. Nesse sentido, Lanna (1999), apud Viegas F^{o} (2000), considera a gestão dos recursos hídricos sendo constituída por uma política de recursos hídricos, por um modelo de gerenciamento de recursos hídricos e por um sistema de gerenciamento de recursos hídricos, que articula as instituições envolvidas e aplica instrumentos legais e métodos para fazer o planejamento de recursos hídricos.

Quanto ao modelo de gestão de recursos hídricos no Brasil, foi adotado o modelo sistêmico de integração participativa, que se caracteriza pela criação de uma estrutura sistêmica, na forma de uma matriz institucional de gerenciamento (Lanna, 1999, Apud Viegas F^{o}, 2000).

Sua adoção foi institucionalizada após a aprovação da Lei 9.433, (BRASIL, 1997) de 8 de Janeiro de 1997, que instituiu a Política Nacional de Recursos Hídricos e criou o Sistema Nacional de Gerenciamento de Recursos Hídricos. Os fundamentos da Lei 9.433 / 97 são os seguintes:

1. A água é um bem de domínio público;
2. A água é um recurso natural limitado, dotado de valor econômico;
3. Em situações de escassez, o uso prioritário dos recursos hídricos é o consumo humano e a dessedentação de animais;
4. A gestão dos recursos hídricos deve sempre proporcionar o uso múltiplo das águas;
5. A bacia hidrográfica é a unidade territorial para implementação da Política Nacional de Recursos Hídricos e atuação do Sistema Nacional de Gerenciamento de Recursos Hídricos;
6. A gestão dos recursos hídricos deve ser descentralizada e contar com a participação do Poder Público, dos usuários e das comunidades.

Quando a propriedade das águas é pública, segundo Lanna (2000), o sistema de gestão de recursos hídricos se caracteriza por três determinações:

1. Surge a necessidade de descentralização da gestão, através da qual o estado, sem abrir mão do domínio sobre a água, permite sua gestão com a participação da sociedade;
2. Adoção de um planejamento estratégico, mediante o qual, governo, usuários de água e sociedade negociam e estabelecem metas de desenvolvimento sustentável atreladas a instrumentos para alcançá-las.
3. Utilização de instrumentos normativos e econômicos, que visam atingir as metas de desenvolvimento sustentável estabelecidas no planejamento estratégico, dentre os quais figuram a outorga dos direitos de uso da água e a cobrança pelo uso dos recursos hídricos, incluindo o lançamento de resíduos nos corpos d’água.

É exatamente nesse contexto que se insere o planejamento e a gestão dos recursos hídricos segundo uma abordagem sistêmica, ou seja, realizar o planejamento estudando o sistema como um todo e preocupando-se com as relações entre suas diversas partes, bem como a utilização de metodologias compatíveis.

2.2 A ANÁLISE SISTÊMICA DE RECURSOS HÍDRICOS E SEUS INSTRUMENTOS

2.2.1 O conceito de Análise Sistêmica de Recursos Hídricos

A abordagem sistêmica caracteriza-se por procurar abordar o assunto em estudo considerando o universo no qual se insere como um sistema, estudando-o como um todo e preocupando-se com as interfaces entre suas diversas partes. Prega o trabalho conjunto de profissionais em equipes interdisciplinares e o estabelecimento de uma linguagem comum entre os diversos especialistas, enfatizando a necessidade de interação e de avaliação permanente (INPE, 1972).

Lanna (1997a) define análise sistêmica de recursos hídricos como uma técnica de solução de problemas complexos de engenharia de recursos hídricos a partir da abordagem sistêmica e do uso de técnicas computacionais agregadas à modelagem matemática. A abordagem sistêmica envolve abstração ou simplificação de um problema complexo, visto de uma forma interligada, como partes de um todo integrado, de tal forma que apenas sejam mantidas as informações mais relevantes para a sua solução.

Segundo Labadie (1987), a aplicação da análise de sistemas e da tecnologia computacional resultarão em redução dos custos operacionais, incremento da eficiência e produtividade, alcance de objetivos considerados anteriormente como inatingíveis e aumento da confiabilidade no sistema.

O homem soluciona problemas a partir de dois elementos essenciais: informações que permitam conhecer uma determinada situação que requer sua atuação e uma concepção intelectual (em geral simplificada) do problema, de quais são suas variáveis, de como elas interagem entre si e com o meio (Porto, 1997). A partir dessa concepção intelectual simplificada pode-se elaborar um modelo do sistema em estudo. Este mesmo autor reafirma de maneira muito apropriada um conceito básico da teoria de modelos, observando que modelar e simplificar são conceitos indissociáveis.

Os sistemas em geral, são representados por modelos. Estes são uma abstração, uma representação simplificada de um sistema, para facilitar o projeto e/ou análise do mesmo. O seu uso ocorre também pela impossibilidade de considerar todas as características e aspectos da realidade.

Loucks (1992) enfatiza o acima exposto ao considerar que os sistemas de recursos hídricos são mais complexos que aquilo que os analistas conseguem ou que, talvez, conseguirão modelar ou programar em seus computadores. As razões não são as limitações computacionais, mas, principalmente, o fato de o homem não compreender suficientemente as múltiplas interdependências entre os processos físicos, bioquímicos, ecológicos, sociais, legais e políticos que governam o comportamento de determinado sistema de recursos hídricos.

As características dos sistemas que são incorporadas nos modelos dependem do que o modelador acha importante ou não. Essas decisões dependem, em última instância, da habilidade e percepção do modelador a respeito do sistema real e de quanto tempo e dinheiro estão disponíveis.

É importante salientar aqui uma observação de Labadie (1987) onde o autor adverte que, no caso de modelos desenvolvidos especificamente para um dado sistema, se o sistema variar fisicamente (pela incorporação de novas estruturas, etc.) ou institucionalmente (pelo reordenamento de prioridades de operação, etc.), o modelo deverá ser modificado para refletir as mudanças e o novo cenário.

Na literatura consultada existem inúmeros autores que apresentam exemplos e casos de aplicação da análise sistêmica em estudos e projetos diversos. Labadie (1987) apresenta técnicas de Análise Sistêmica e sua aplicação ao caso dos estudos sobre operação e segurança do sistema de reservatórios de Valdesia na República Dominicana. Goulter (1992) apresenta o uso de técnicas de análise sistêmica para projetar redes de distribuição de água.

2.2.2 Principais Metodologias usadas na Análise Sistêmica de Recursos Hídricos

Segundo Simonovic (1993), as diferentes aplicações da abordagem sistêmica no planejamento e operação de sistemas de recursos hídricos podem ser classificadas em:

a) Simulação;
b) Otimização;
c) Análise multiobjetivo;
d) Combinações das técnicas anteriores.

2.2.2.1 Simulação

Um modelo de simulação tenta reproduzir o comportamento de um determinado sistema, como, por exemplo, uma bacia hidrográfica, através de algoritmos, ou seja, tenta reproduzir as respostas desse sistema a um conjunto de informações de entrada. As informações de entrada podem incluir regras de decisão, permitindo ao decisor examinar as conseqüências de diversos cenários de um sistema existente ou em fase de projeto.

Resumidamente poderia-se dizer que os modelos de simulação tentam responder à pergunta “o que aconteceria se...?”.

A simulação não é capaz de gerar diretamente uma solução ótima de um problema mas pode ser uma interessante maneira de testar distintas alternativas de decisão e detectar um ótimo (provavelmente não global) por enumeração.

Os conceitos de base da simulação em geral são fáceis de entender. Daí a vantagem que tem os modelos de simulação sobre os outros tipos de análises de sistemas hídricos.

De acordo com Simonovic (1993), modelos típicos de simulação associados com operação de reservatórios incluem balanços de massa, computando vazões afluentes e efluentes e diferenças de armazenamento. Eles podem incluir valoração econômica de danos causados por cheias, benefícios pela geração de energia e/ou irrigação, entre outros. Geralmente os modelos de simulação permitem um bom detalhamento e representação da realidade do complexo físico e econômico, bem como das características sociais envolvidas em um sistema de reservatórios.

A simulação, segundo Braga Jr. (1987), procura somente representar um sistema em seus aspectos importantes, permitindo, por exemplo, o acompanhamento dos níveis dos reservatórios, a determinação do valor de funções objetivo ao longo do tempo, dentre outras possibilidades. A vantagem dos modelos de simulação reside no fato de permitirem uma representação relativamente detalhada do projeto e, dessa maneira, avaliar com maior precisão o comportamento do sistema projetado.

A simulação é muito flexível e amplamente utilizada. Sua flexibilidade permite que praticamente qualquer sistema seja representado matematicamente através de modelos para computador.

A grande desvantagem é que a simulação não oferece aos usuários a oportunidade de restringir o espaço decisório e, por conseqüência, a solução de problemas é alcançada através do exaustivo processo de tentativa e erro.

Pode-se optar em utilizar um modelo de simulação genérico, aplicando-o a casos semelhantes, ou um específico, desenvolvido para uma situação peculiar em análise (Luz, 1994). Tal escolha dependeria dos objetivos da análise, da potencialidade dos modelos disponíveis, dos dados disponíveis, dos custos e do tempo envolvidos (Zahed, 1987 apud Luz, 1994).

Uma observação importante é feita por Yeh (1985), ao afirmar que nos anos mais recentes a tendência tem sido incorporar esquemas de otimização dentro de modelos de simulação para realizar certos níveis de otimização. Como resultado disto, a distinção geralmente feita entre a otimização e a simulação tende a ser obscurecida.

Neste trabalho serão desenvolvidas ferramentas para um modelo de simulação.

2.2.2.2 Otimização

Braga Jr (1987) afirma que, em termos práticos, os métodos de otimização são métodos que permitem, tanto no planejamento quanto na operação, tomar decisões que são ótimas em algum sentido mensurável. Porém, é importante acrescentar que tais decisões são ótimas sob o ponto de vista matemático.

Um modelo de otimização é constituído por uma “função objetivo” que se deseja maximizar ou minimizar (otimizar):

$$FO = Max / Min \{ F(X1, X2, ..., Xn)\} \qquad \text{(Eq. 2.1.)}$$

e onde devem ser satisfeitas algumas “restrições”:

$$gi(X1, X2, ..., Xn) > bi; < bi; = bi$$

onde: *Xj* - variáveis de decisão (j = 1, 2, ..., n)
gi - funções de restrição (i = 1, 2, ..., m)
bi - parâmetros do modelo

As soluções que satisfazem as restrições definem o conjunto das soluções viáveis. Dentre as soluções viáveis, a que maximiza ou minimiza a função objetivo, conforme o caso, é chamada de solução ótima.

No caso de os parâmetros e/ou variáveis serem considerados conhecidos e com seus valores definidos o modelo chama-se de programação determinística, enquanto que se os mesmos forem considerados variáveis aleatórias, obedecendo a uma distribuição estatística, tem-se uma situação de programação estocástica.

As restrições obrigam o modelo a seguir leis físicas e determinações econômicas, sociais, legais, dentre outras. São exemplos de restrições típicas em sistemas de reservatórios, a equação de conservação de massa e o armazenamento máximo.

Os pontos de ótimo que se busca atingir – dentro de uma visão puramente matemática - são pontos estacionários, ou seja, pontos de máximo/mínimo locais ou globais.

Na determinação do ótimo global, o conceito de combinação convexa é fundamental, ou seja, quando dois pontos quaisquer de uma função objetivo podem ser unidos por um segmento e este segmento fica totalmente inserido em um semi-espaço definido por essa função. Nessas condições pode-se afirmar que, caso a F.O. seja convexa, o mínimo local é também o mínimo global e, no caso da F.O. ser côncava, o máximo local será o máximo global (Braga Jr., 1987). Os conceitos de convexidade e concavidade também são estendidos para o caso de n variáveis.

A principal desvantagem da otimização é que, quase sempre, requer um modelo de estrutura simplificada, podendo significar um desvio da realidade. Porém, os administradores de recursos hídricos sentem-se mais atraídos por modelos que permitam incorporar a realidade da melhor forma possível, embora a convergência para o ótimo global não seja assegurada (Labadie, 1987).

Goulter (1992) relata que uma revisão sobre análise sistêmica revela que as aplicações em geral são realizadas por pesquisadores, havendo pequena ou nenhuma aceitação das técnicas de otimização na prática de engenharia de recursos hídricos. Coloca que, talvez, a causa dessa pequena aceitação seja em função de alguns modelos teóricos terem sido desenvolvidos com representações muito simplificadas da realidade, ou pelo fato de que os modelos teóricos, quando muito precisos na representação da realidade, podem ser de complicada e difícil aplicação.

A otimização dos componentes de um projeto, individualmente, nunca dará maiores benefícios que a otimização do sistema como um todo. No entanto existem duas dificuldades principais para se atingir tal objetivo:

a) à medida que o número de componentes do sistema aumenta, os requerimentos computacionais podem ser incrementados ao extremo;

b) a coleta e o processamento de um grande volume de dados podem exceder as restrições de tempo, orçamento e possibilidade de erros.

A definição dos objetivos e a formulação de medidas quantitativas são duas etapas fundamentais na aplicação da análise sistêmica. A primeira envolve o chamado decisor, ou seja, o grupo a quem cabe a tomada de decisões. A segunda refere-se à modelagem, no seu aspecto geral, e à definição matemática, como tradutora dessa modelagem para o alcance dos objetivos estabelecidos na primeira etapa. A representação matemática desses objetivos é chamada de função objetivo.

Segundo Wurbs (1993), a maioria das funções objetivo descritas na literatura são de benefício e custo econômicos, disponibilidade e confiabilidade na oferta de água e/ou geração de energia hidroelétrica. Um modelo de otimização geralmente incorpora uma única função objetivo. Múltiplos objetivos podem ser combinados, também, através de uma expressão única quando for possível sua medição em uma unidade comum.

Conforme Simonovic (1993), a maioria dos modelos de otimização são baseados em alguma técnica de programação matemática. Sua classificação básica é a seguinte:

a) Programação Linear (PL);

b) Programação Dinâmica (PD); e

c) Programação Não-Linear (PNL).

Todas essas técnicas podem ser aplicadas em ambientes determinísticos ou estocásticos e possuem características comuns, tais como necessitar de uma função objetivo, ter ou não restrições e variáveis de decisão.

2.2.2.3 Análise Multiobjetivo

A análise multiobjetivo procura quantificar os processos de tomada de decisão com múltiplos objetivos (Braga Jr. e Gobetti, 1997).

A utilização de uma abordagem multiobjetivo tem algumas vantagens sobre as técnicas convencionais com objetivos únicos, como poder incorporar objetivos não mensuráveis ou permitir que as funções objetivo sejam medidas em diferentes escalas ou unidades. Além disso, é possível avaliar, através de funções de intercâmbio, quanto melhora um dos objetivos enquanto outros pioram.

Braga Jr. e Gobetti (1997) citam como vantagem dessas técnicas o fato de permitir o conhecimento explícito das relações existentes entre funções objetivo conflitantes, comuns em situações onde é necessário alocar recursos escassos a diferentes atividades.

Ocorre que o tradicional conceito de otimização, onde se busca o máximo ou o mínimo de uma função objetivo, encontra uma dificuldade importante na análise multiobjetivo: a não existência de um único ótimo em um problema com múltiplos objetivos. O que existe, segundo Braga Jr. e Gobetti (1997), é um conjunto de ótimos que satisfazem de formas diferentes, os diferentes objetivos envolvidos na análise.

Uma análise multiobjetivo seria feita basicamente em duas etapas. Na primeira, seria realizada uma enumeração das soluções dominantes, também chamadas de soluções de Pareto, ou seja, daquelas soluções cuja melhora, em relação a um dos objetivos, só pode ser obtida com a piora, em relação a outro objetivo. A segunda etapa envolve uma certa subjetividade e nela seria feita a escolha de um ponto, dentro do conjunto de soluções dominantes, considerado pelos decisores como satisfatório aos objetivos do trabalho, também chamada de melhor solução de compromisso.

Aqui surge o conceito de “Compromisso”, ou “Trade off”. Esse conceito poderia ser traduzido basicamente como a negociação da quantidade de uma função objetivo que precisaria ser sacrificada para obter ganhos adicionais em outra das funções objetivo.

Braga Jr. e Gobetti (1997) apresentam várias técnicas e métodos aplicados a análise multiobjetivo, com respectivos exemplos, desenvolvidos por diversos autores.

2.2.2.4 Considerações Finais Sobre a Utilização de Metodologias de Análise Sistêmica na Área de Recursos Hídricos

Diante dessa revisão de metodologias, verifica-se a importância e a necessidade da abordagem sistêmica no planejamento de sistemas de recursos hídricos. O uso múltiplo da água é uma realidade nítida que os projetos futuros deverão considerar.

É importante enfatizar novamente que, no planejamento de recursos hídricos, cada caso é único e que as suas características indicam quais técnicas de análise sistêmica são mais adequadas para aplicar ao caso em estudo.

Pode-se indicar a simulação como uma das técnicas para aplicação de uma abordagem sistêmica, com a vantagem de ser extremamente flexível, com conceitos básicos de fácil entendimento, não sendo, porém, capaz de gerar diretamente uma solução ótima. Por outro lado, pode-se indicar a otimização utilizando a programação linear, quando aplicável, onde esta tem as vantagens de garantir um ótimo global, não ter problemas de dimensionalidade e para a qual existem pacotes computacionais gerais prontos; sabe-se, entretanto, que esta apresenta as mencionadas limitações de que a função objetivo e as restrições devem ser lineares. Cabe ao profissional a escolha das ferramentas mais adequadas a cada caso.

É comum, também, a utilização de modelos de otimização por programação linear, com todas as simplificações que são inerentes à linearização de problemas complexos, combinada com modelos de simulação, onde estes têm o propósito de examinar os resultados da otimização.

Por fim, ao procurar resolver um problema de planejamento de recursos hídricos, o profissional deve buscar as ferramentas adequadas sem desconsiderar a possibilidade de utilizar não apenas uma, mas um conjunto de ferramentas e/ou técnicas, procurando dessa forma melhor aproveitar as vantagens de cada uma.

2.2.3 Incertezas Envolvidas na Análise Sistêmica de Recursos Hídricos

Independente da metodologia adotada para realizar estudos com enfoque sistêmico, um fator importante no estudo de recursos hídricos é a incerteza do futuro.

Sabe-se que a água distribui-se de modo irregular no tempo e no espaço. Ocorre que essas variações sazonais e multianuais podem ser bastante significativas, de modo que o potencial economicamente explorável e que deve ser considerado no planejamento, pode ser bem menor do que o potencial indicado pelas vazões médias de longo período.

A incerteza do futuro é a maior dificuldade com que qualquer processo decisório se defronta, pois tudo o que se conhece é o passado e tudo o que importa no processo decisório é o futuro. Nos sistemas de Recursos Hídricos existem, além da aleatoriedade dos processos hidrológicos associados à disponibilidade hídrica, aquelas aleatoridades associadas aos problemas econômicos, sociais e ambientais que determinam as demandas de água. Segundo Lanna (1997a), é prática usual em análise sistêmica dos recursos hídricos trabalhar-se diretamente apenas com a incerteza hidrológica, considerando os demais tipos de incerteza através de processos de cenarização. O autor divide as abordagens no tratamento da incerteza hidrológica em duas grandes classes, de acordo com a maneira como esta aleatoriedade do futuro é inserida.

Abordagem explicitamente estocástica: inserem os modelos probabilísticos que descrevem a aleatoriedade do futuro na formulação do problema decisório. A regra decisória é produzida de forma direta como resultado do processo computacional, associando as decisões que são buscadas ao estado do sistema.

Abordagem implicitamente estocástica: o problema decisório é resolvido supondo que os eventos hidrológicos futuros são conhecidos - ou adotando a própria série hidrológica amostral disponível ou, ainda, de forma mais refinada, desenvolvendo modelos estocásticos de simulação dessas séries hidrológicas (geração de séries sintéticas). O problema decisório seria resolvido tantas vezes quantas fossem as séries sintéticas geradas. Para cada série hidrológica utilizada é produzida uma seqüência temporal de decisões, ótima para esta série. A regra decisória deve ser produzida em uma segunda etapa, através de uma pesquisa que associe uma lógica decisória ao estado do sistema (através de gráficos e/ou realização de regressões múltiplas para ajuste de uma função matemática à relação que exista entre as variáveis decisórias e as que identificam o estado do sistema analisado).

2.3 SIMULAÇÃO APLICADA A SISTEMAS DE RECURSOS HÍDRICOS

2.3.1 Fundamentos da Simulação de Sistemas de Recursos Hídricos baseados em Redes Hidrográficas

Observou-se que a simulação é uma das técnicas mais flexíveis e eficientes para a aplicação de uma abordagem sistêmica na área de recursos hídricos. Para a sua aplicação a um Sistema de Recursos Hídricos, o primeiro passo é a abstração das principais característica e elementos que compõe o sistema real.

Pode-se considerar o sistema real como um conjunto composto por um Sistema Físico Natural, Estruturas de Controle, um conjunto de Demandas e um Sistema Institucional de Operação e Controle.

O Sistema Físico Natural corresponde à representação do Ciclo Hidrológico e contém todos os elementos que integram entre si os processos atmosféricos e aqueles que ocorrem na Bacia Hidrográfica, responsáveis pela disponibilidade hídrica (Viegas Fº, 2000), como por exemplo à evaporação potencial, a precipitação, os canais naturais, a superfície da bacia hidrográfica, etc.

As estruturas de controle correspondem aos reservatórios, diques, pontos de medição ou pontos de controle e às tomadas d'água para o atendimento de demandas, existentes ao longo da bacia hidrográfica.

As próprias demandas existentes na bacia compõem o sistema real, sendo de diversos tipos e com diferentes características de distribuição espaço-temporal, como por exemplo demandas difusas, urbanas, de irrigação, etc.

Compondo o sistema real, pode-se, ainda, considerar a presença de um sistema institucional de operação e controle, que corresponde às limitações legais, a decisões de planejamento estratégico e de operação tática.

Em um segundo momento, é necessário representar os principais elementos e característica que foram identificados no sistema real. A utilização de Redes hidrográficas é uma forma de reproduzir com razoável aproximação a condição natural de uma bacia hidrográfica, com sua superfície e seus canais, ao longo do qual a água é propagada e pode ser captada para uso.

A Figura 2.1 apresenta um exemplo de organização esquemática de uma Rede Hidrográfica aplicada a um Sistema de Recursos Hídricos. A figura contém uma rede composta por Nós (de passagem ou de armazenamento), que constituem os chamados Pontos de Controle ou Pontos Característicos (PCs) e pelos Trechos de Água que os unem, lançados sobre um esquema de Bacia Hidrográfica (Viegas Fº, 2000).

Esse esquema apresenta, entre outros itens, as divisões entre as sub-bacias, formando um polígono que corresponde à definição topológica de sub-bacia, o exutório de cada Sub-Bacia, o limite da Bacia e o Ponto Característico que lhe fica mais à jusante, no exutório da mesma.

Também são apresentados nessa figura os cursos de água que cortam as Sub-Bacias e que sejam importantes de serem destacados, os quais se confundem, no exemplo, com os Trechos de Água que estão ali colocados para representá-los. O termo Trecho de Água, no atual contexto, corresponde à representação de um trecho de curso de água, natural ou artificial, unindo dois Pontos Característicos.

Os Pontos Característicos são locais ao longo da bacia hidrográfica que apresentam importância e que portanto merecem receber algum tipo de controle. É possível identificar-se PCs, por exemplo, em exutórios de sub-bacias, em reservatórios de armazenamento, em pontos de geração de energia, em captações de demandas hídricas localizadas ou, ainda, em simples pontos detecção de vazão.

Observa-se que a bacia hidrográfica é considerada como sendo composta por sub-bacias unidas por cursos d'água, em cujas extremidades encontram-se os pontos característicos, que representam, de forma segmentada, a rede de drenagem (Viegas Fº, 2000).

Uma vez definida a representação do sistema pela rede hidrográfica, deve-se propor um modelo que simule a dinâmica da propagação da água ao longo dessa rede, de forma que cada PC receba as contribuições de sub-bacias e possíveis PCs a montante, seja processada a transformação da chuva em vazão, sejam atendidas as demandas difusas ou localizadas em cada PC e que a água remanescente seja propagada aos PCs de jusante.

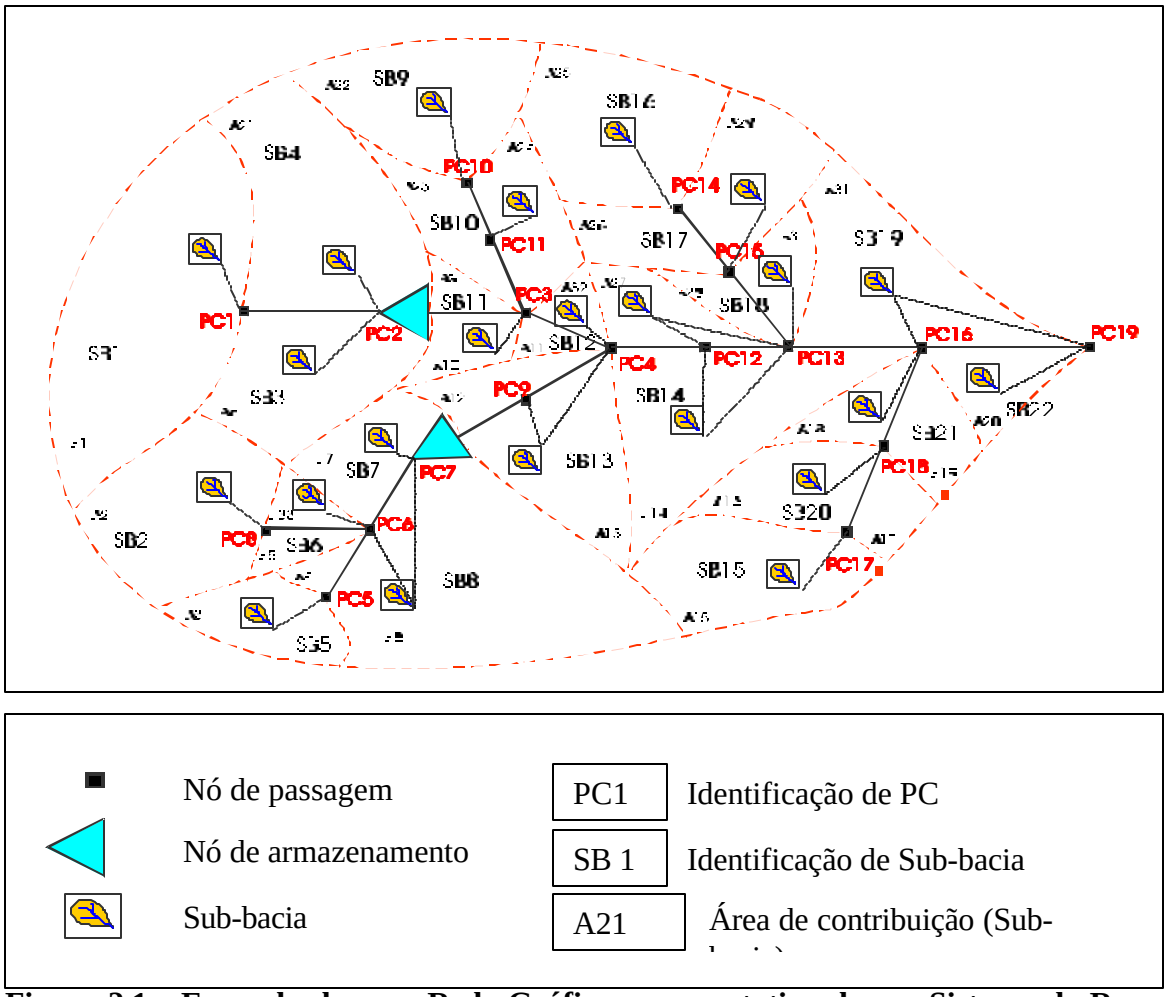


Figura 2.1 - Exemplo de uma Rede Gráfica representativa de um Sistema de Recursos Hídricos. (Fonte: Viegas Fº, 2000).

Porém, para que o sistema real seja representado o mais fielmente possível, é necessário que exista a possibilidade de ser simulada a presença de um sistema institucional de operação e controle, que poderia intervir no sistema através de decisões de planejamento estratégico e de operação tática.

Dessa forma é possível, através da simulação baseada em redes hidrográficas, testar métodos de controle do uso da água (estratégias operacionais) através de estruturas (reservatórios) e suas respectivas regras operacionais e/ou através de medidas disciplinadoras do uso da água (racionamento).

2.3.2 Simulação de Estratégias Operacionais de Planejamento do Uso da Água

Uma estratégia procura estabelecer decisões em situações de incerteza do futuro, baseadas no estado corrente do sistema e nas informações existentes que permitam prever as condições de operação futuras (Lanna, 1982).

Ao longo do tempo, foram desenvolvidas e testadas várias estratégias para a operação de reservatórios, nos mais diferentes sistemas de recursos hídricos. Alguns autores também as denominam regras operacionais. Algumas dessas regras operacionais serão aqui abordadas.

Dentre as possíveis estratégias ou regras para a operação de reservatórios, provavelmente a mais simples é a que estabelece defluências fixas ao longo do ano, sem nenhum tipo de controle sobre as demandas.

A regra de defuência fixa poderia ser aplicada, por exemplo, em situações onde houvesse um acordo ou contrato entre usuários de água e uma instituição responsável pela operação de um reservatório, para o fornecimento de água em quantidades fixas ao longo de determinados períodos do ano.

Porém, a aplicação dessa regra, simplesmente, só será viável em situações onde não haja o risco das demandas não serem atendidas, pois nesse caso deverá ser aplicado algum tipo de racionamento às demandas. A forma mais básica para estabelecer um possível racionamento é a chamada Regra Padrão de Decisão (Lanna, 1993, apud Vianna Jr., 1998).

A regra padrão de decisão para operação de reservatórios determina o atendimento das demandas enquanto houver água disponível, ou seja, caso a quantidade de água disponível no reservatório seja maior ou igual às demandas, estas serão atendidas integralmente; caso as demandas previstas superem a disponibilidade hídrica, será liberada a quantidade de água que estiver disponível no reservatório, promovendo o atendimento parcial das demandas. Essa regra prevê, ainda, a possibilidade de vertimento de água do reservatório, caso a disponibilidade hídrica supere a sua capacidade máxima.

Existe uma regra operacional que foi elaborada com base na Regra Padrão, tendo como uma das principais vantagens operacionais a sua simplicidade. É denominada Regra Padrão Modificada (Lanna, 1993 apud Vianna Jr., 1998).

A Regra Padrão Modificada adota um sistema de racionamento preventivo (com antecipação do racionamento), onde nem toda a água disponível será imediatamente utilizada. Somente quando houver uma certa margem de segurança, no que se refere à disponibilidade hídrica, é que a demanda será integralmente atendida. O rigor do esquema de racionamento pode ser definido no momento da implementação da regra operacional.

A Regra Padrão Modificada também exerce controle sobre o volume de espera do reservatório. Quando a disponibilidade hídrica for superior a um determinado valor, definido pela regra, será liberado um volume de água superior à demanda. Dessa forma, pode-se variar o volume de espera do reservatório que estiver sendo operado.

Uma outra regra de operação é que estabelece níveis ou volumes estratégicos em reservatórios, utilizados para atender demandas de prioridades diferentes. Esses níveis definem zonas intermediárias, nas quais as demandas estão submetidas a racionamento, admitindo-se apenas duas situações: - atendimento ou não-atendimento.

Segundo Lanna (1982), o conceito de zoneamento de reservas foi apresentado por Beard, em 1967 e se aplica à operação de sistemas com múltiplos reservatórios. Vários critérios operacionais podem ser adotados, tendo por base prioridades ou outras relações entre os reservatórios do sistema.

Dentre outras existentes, pode-se citar, ainda, as regras-guias ou curvas-guias para operação de reservatórios. São regras que definem a estratégia de operação de reservatórios ao longo do tempo através de gráficos, equações ou conceitos (Lanna, 1982).

Pode-se ter, por exemplo, curvas que indiquem os volumes máximos e mínimos para um dado reservatório ao longo do ano, evitando que ocorram falhas graves no atendimento de demandas futuras e, ao mesmo tempo, evitando danos que poderiam ser provocados por cheias.

2.4 O MODELO PROPAGAR MOO

2.4.1 Aspectos Gerais

O PROPAGAR é um modelo que simula a propagação de vazões ao longo de um sistema hídrico, representado por uma rede hidrográfica. Este modelo é parte integrante de um sistema maior denominado SAGBAH – Sistema de Apoio ao Gerenciamento de Bacias Hidrográficas.

O SAGBAH constitui-se por um Sistema de Apoio à Decisão orientado para as atividades de Planejamento e Gestão de Recursos Hídricos, desenvolvido no Instituto de Pesquisas Hidráulicas da Universidade Federal do Rio Grande do Sul, pelo Prof. Antônio Eduardo Lanna com a participação e colaboração, ao longo do tempo, de alguns dos seus orientados no Programa de Pós-graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental (Viegas F^{o}, 2000).

O SAGBAH é composto por um conjunto de programas de modelagem matemática para diversas finalidades, inicialmente desenvolvidos para a plataforma DOS, em linguagem FORTRAN. Atualmente, novas versões dos modelos, bem como novos aplicativos, estão sendo desenvolvidos segundo o paradigma da Modelagem Orientada a Objetos, utilizando-se a Linguagem Borland Object Pascal, sob coordenação dos professores Antônio Eduardo Lanna (UFRGS) e João Soares Viegas F^{o} (UFPel).

Dentre os módulos integrantes do SAGBAH, os que já estão desenvolvidos são o CHUVAZ 2000, o MODHAC 2000, o CASCATA 2000 e o PROPAGAR 2000 (Viegas F^{o} e Lanna, 1999). A nova versão do modelo Propagar, desenvolvida segundo a Modelagem Orientada a Objetos, é aqui identificada como PROPAGAR MOO e sua concepção básica mostrada a seguir.

2.4.2 A concepção básica do Modelo PROPAGAR MOO

O modelo PROPAGAR MOO simula a propagação de vazões em uma bacia hidrográfica, submetida a decisões operacionais referentes ao suprimento de demandas hídricas e à operação de reservatórios.

Para permitir a simulação da propagação de vazões, o modelo baseia-se em uma estrutura de Rede Hidrográfica, sobre a qual são definidos pontos característicos ou pontos de controle (aqui chamados simplesmente de PCs). Os PCs são unidos por segmentos que representam a rede de drenagem da bacia hidrográfica, conforme demonstra a Figura 2.2.

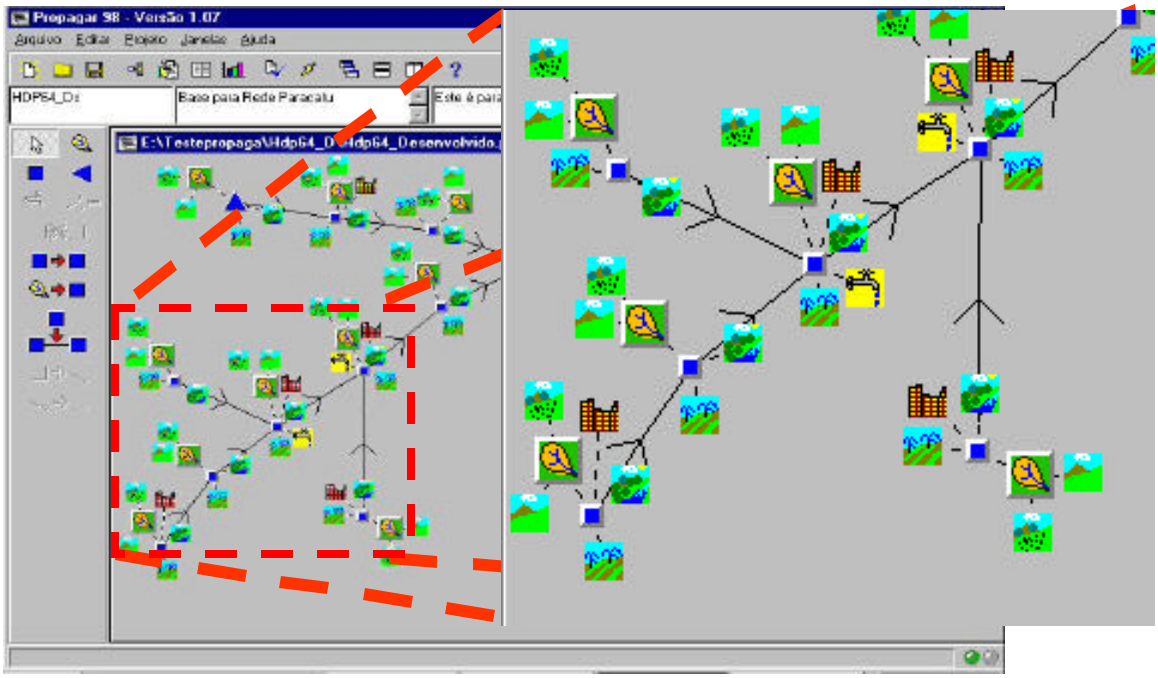


Figura 2.2 – Área de projeto do Propagar MOO. (Fonte: Viegas Fº et. al., 2001).

Na verdade, o modelo adota a concepção de sistema dentro da perspectiva da Modelagem Orientada a Objetos, onde o sistema é visto como um conjunto de objetos que, inter-relacionados e interagindo entre si, reproduzem as características que se deseja abstrair do sistema real (adaptado de Viegas Fº, 2001). Nessa perspectiva, por exemplo, o modelo trata cada PC como um objeto, dotado de características próprias, representadas por atributos e por métodos a ele aplicáveis.

A determinação da quantidade e localização dos PCs é feita de forma que sejam representados os pontos onde se localizam demandas significativas (ou soma de demandas), saídas de reservatórios e exutórios de sub-bacias, ficando a critério do usuário a definição de cada PC. Cada um dos PCs pode estar submetido ao controle de um reservatório e recebe afluências hídricas de, pelo menos, uma sub-bacia incremental - resultantes do processo de transformação de chuva em vazão - e afluências hídricas originadas dos PCs à montante. Cada sub-bacia pode estar ligada, por sua vez, a mais de um PC, desde que adjacentes.

As demandas hídricas do sistema, representadas na rede, podem estar ligadas a PCs (demandas localizadas) ou a Sub-bacias (demandas difusas). As demandas ligadas a PCs (demandas localizadas) são agrupadas em Classes de Demandas e são divididas em três ordens de prioridades: primárias, secundárias e terciárias. Isto estabelece uma relação de preferência no atendimento das demandas.

Para estabelecer o relacionamento entre os PCs e sistematizar a propagação das vazões ao longo da bacia, em cada intervalo de tempo, o modelo utiliza uma classificação hierárquica dos Pontos Característicos. Dessa forma, os PCs que não recebem contribuição de nenhum outro são considerados de ordem 1, os que recebem contribuições de PCs de ordem 1 são considerados de ordem 2 e, assim por diante até que todos os PCs estejam classificados. Isso para que, ao se fazer a análise em um PC de ordem hierárquica superior, as afluências vindas de PCs de montante (de menor hierarquia) já tenham sido calculadas.

O Propagar MOO também incorpora novas características, mantendo a concepção básica de simulação do modelo em sua versão para plataforma DOS. Dentre as novas características, destacam-se as seguintes (adaptado de Viegas F^{o}, 2000):

- Interface com editor gráfico de Rede Hidrográfica através do uso de uma Área de Projeto e de uma Barra de Ferramentas Hidrológicas (Figura 2.2);
- Gerenciador de Projetos que possibilita total controle sobre todos os elementos criados na Rede Hidrográfica;
- Gerenciamento de criação, controle e edição de Demandas (primárias, secundárias e terciárias), reunindo-as em Classes de Demandas;
- Possibilidade de definição, por parte do usuário, de diferentes Unidades de Consumo de Água e de Unidades de Demandas;
- Disponibilização dos resultados na forma de arquivos ASCII (.txt) e Planilhas de Cálculo padrão Excel (.xls).

No que diz respeito às decisões operacionais, estas podem ser introduzidas pelo usuário em dois momentos. No primeiro, considerado de planejamento estratégico, são estabelecidas as políticas operacionais para todos os PCs. No segundo momento, considerado de operação tática, verifica-se a possibilidade das decisões estratégicas serem implementadas.

Para que o usuário possa implementar as suas estratégias e táticas operacionais, existe a possibilidade da utilização de “scripts”. Estes, conforme já mencionado, são pequenas rotinas de programação que utilizam bibliotecas do próprio modelo para adequar o mesmo às necessidades de cada problema a ser resolvido.

2.4.3 A Dinâmica de Simulação

O objetivo do modelo Propagar MOO, conforme já apresentado, é o de simular a propagação de vazões através da bacia hidrográfica, de montante para jusante, ao longo do tempo, buscando atender demandas existentes em face das disponibilidades hídricas da bacia. Para atingir o objetivo, é realizado em cada ponto característico, a cada intervalo de tempo, um balanço hídrico que considera a afluência de água proveniente dos PCs de montante e das suas próprias sub-bacias, bem como as demandas difusas e localizadas que devem ser supridas. A Figura 2.3 apresenta as equações de balanço hídrico do Propagar MOO.

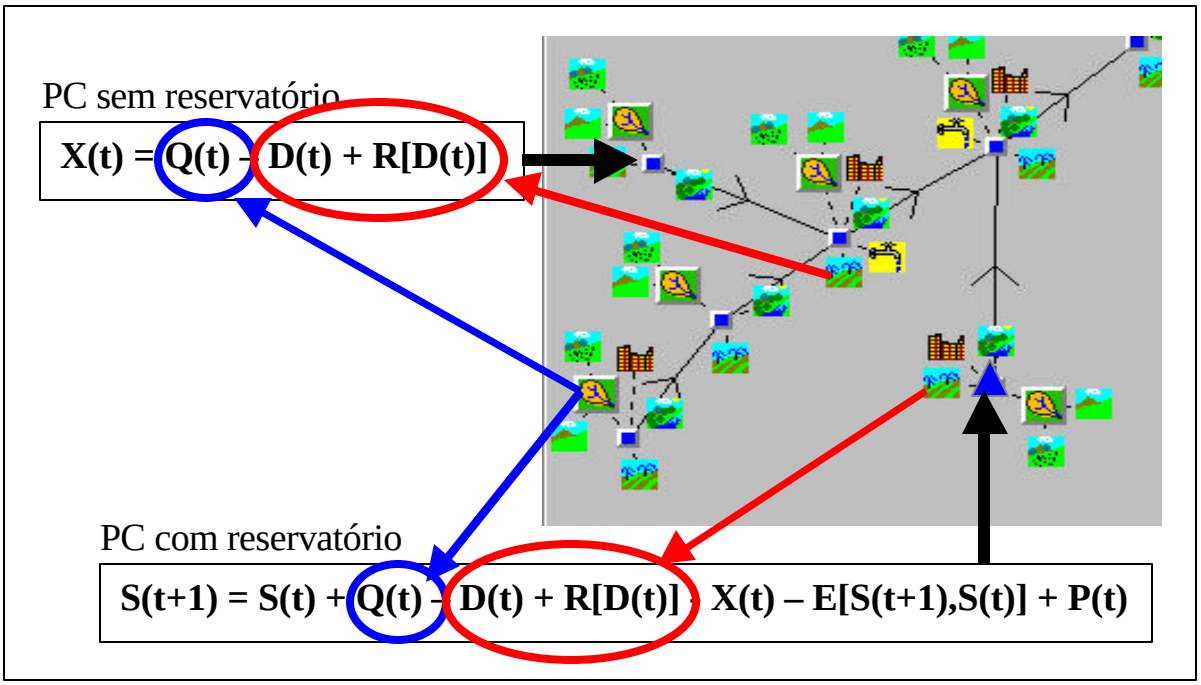


Figura 2.3 – Equações de Balanço Hídrico do Propagar MOO.

Quando o PC corresponder a um nó sem a presença de um reservatório, a equação de que representa de balanço hídrico será:

$$X(t) = Q(t) - D(t) + R[D(t)] \qquad \text{(Eq. 2.2.)}$$

Onde:

X(t) – Representa a vazão defluente do PC no intervalo de tempo t;

Q(t) – Representa a contribuição afluente total (contribuição das sub-bacias laterais, acrescida da contribuição dos PC's de montante);

D(t) – Representa o total das demandas concentradas a serem supridas nesse PC;

R[D(t)] – Representa o retorno de água que possa lhe corresponder (Lanna, 1997c).

No caso de PCs controlados por reservatórios a equação será a fórmula clássica de balanço hídrico em reservatórios:

$$S(t+1) = S(t)+Q(t)-D(t)+R[D(t)] -X(t)-E[S(t+1),S(t)] \qquad \text{(Eq. 2.3.)}$$

Onde:

S(t) – Representa o armazenamento no reservatório no início do intervalo de tempo t;

E[t] – Representa a evaporação correspondente a *S(t)*;

S(t+1) – Representa o armazenamento ao final do intervalo de tempo;

X(t) – Representa uma vazão defluente, destinada ao atendimento de demandas à jusante controlável pela política operacional do reservatório, estabelecida pelo usuário.

As demandas difusas são totalizadas e atendidas no âmbito da própria sub-bacia à qual estão ligadas, quando existir água para tal. Dessa forma, as vazões afluentes ao PC, durante um determinado intervalo de tempo, oriundas do processo de transformação chuva-vazão ocorrido nas próprias sub-bacias que o alimentam, já deverão ser descontadas das demandas difusas que ali terão de ser supridas. (Viegas F^{o}, 2000).

A seguir demonstra-se a equação que representa essa operação (Viegas Fº, 1999c).

$$Q_{sb}(t) = \sum_{Sbs} cc_i [QD_i(t) - \sum_j DD_j(t)] \qquad \text{(Eq. 2.4.)}$$

$$DD_j(t) = FC.UCA_j(t).UD_j(t).AREA_{SB}.ED.FI(t) \qquad \text{(Eq. 2.5.)}$$

onde:

$Q_{sb}(t)$ Vazão afluente total das sub-bacias que contribuem a um determinado PC, em m^3/s, no intervalo de tempo *t*, já descontadas as demandas difusas;

$QD_i(t)$ Vazão, em m^3/s, oriunda de processo difuso de transformação chuva-vazão, na Sub-bacia *i*, durante o intervalo de tempo *t*;

$DD_j(t)$ Demanda Difusa *j*, em m^3/s, a ser atendida na Sub-bacia *i*, durante o intervalo de tempo *t*;

Cc_i Coeficiente de Contribuição da Sub-bacia *i* para o PC considerado;

$UCA_j(t)$ Valor, em Unidades de Consumo de Água, da demanda *j* no intervalo de tempo *t*;

FC Fator de Conversão para a classe de demanda à qual a demanda pertence. (transforma a UCA para m^3/s);

UD(t) Nº de Unidades da Demanda *j* que deverão ser atendidas no intervalo de tempo *t*, no PC, na sub-bacia;

$AREA_{sb}$ Área da sub-bacia;

ED Escala de Desenvolvimento;

FI(t) Fator de Implantação.

As demandas localizadas (ligadas diretamente aos PCs) são totalizadas em três tipos, quanto à sua prioridade de atendimento: demanda primária, demanda secundária e demanda terciária. A formulação matemática que representa essa totalização é a seguinte:

$$DL_j(t) = FC.UCA_j(t).UD_j(t).ED.FI(t) \qquad \text{(Eq. 2.6.)}$$

onde:

$DL_j(t)$ Demanda Localizada j, em m^3/s, a ser atendida em um certo PC, durante o intervalo de tempo t.

As demais notações mantêm-se inalteradas.

Como o objetivo deste tópico é fazer uma revisão da dinâmica de simulação do modelo, não serão abordados aqui, com profundidade, os elementos das equações acima. Para tal, indica-se Viegas Fº (2000), onde todos os elementos das equações são esclarecidos com detalhes.

No que se refere aos intervalos de tempo para a simulação, o modelo permite que estes possam ser de 5 dias, 10 dias, 15 dias ou 30 dias (mensal), considerando, porém, que as vazões afluentes ao PC localizado mais à montante, na bacia hidrográfica, possam atingir o trecho mais à jusante, dentro do intervalo de tempo de simulação definido pelo usuário.

As decisões gerenciais são introduzidas em duas fases. Na primeira fase, denominada fase de planejamento estratégico, são estabelecidas as políticas operacionais para todos os PCs em função da água existente na bacia naquele momento. Isso é feito considerando-se conhecer a água que afluirá ao PC durante o intervalo de tempo. Nos cursos de água sem controle de reservatório a política a ser aplicada - e testada pelo modelo - refere-se ao nível de atendimento das demandas hídricas supridas no trecho. Nos cursos de água com reservatório, a estratégia operacional envolve o estabelecimento do nível de atendimento à demanda e a descarga defluente do reservatório (Lanna, 1997d).

Na fase seguinte, de operação tática, verifica-se a possibilidade das decisões estratégicas serem implementadas. Aí são confrontados restrições e condicionamentos de origem física, como, por exemplo, a existência de água para atendimento a uma demanda ou descarga, ou de origem gerencial, como se haverá de fato racionamento não obstante haver água para suprimento a uma demanda (Lanna, 1997d).

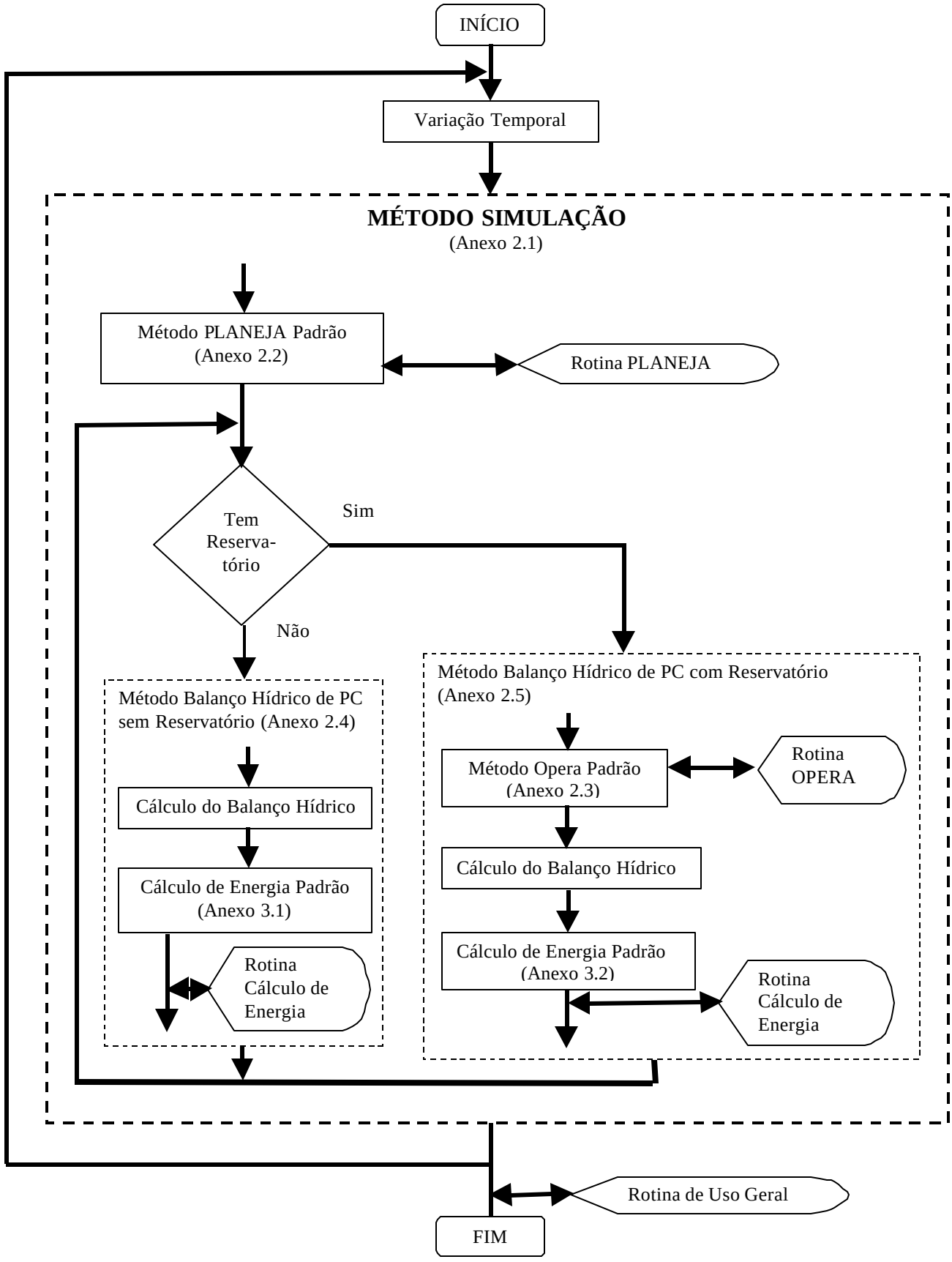


Figura 2.4 – Fluxograma Básico do Propagar MOO.

Para melhor compreensão da dinâmica do modelo e das fases para introdução das decisões gerenciais, a Figura 2.4 apresenta o fluxograma básico do modelo Propagar MOO, identificando seus principais métodos e as chamadas de rotinas criadas pelo usuário.

Para introduzir a estratégia operacional usa-se método denominado Planeja (Anexo 2.2) e para introduzir a tática operacional usa-se um outro método chamado Opera (Anexo 2.3). O primeiro serve para um planejamento geral, enquanto que o segundo, para a operação de reservatórios. Esses métodos são implementados com alternativas padronizadas (Planeja Padrão e Opera Padrão), mas o modelo permite modificá-los, caso o usuário deseje.

As decisões operacionais modificadas pelo usuário são introduzidas através de "scripts" de planejamento e operação, denominados Rotina PLANEJA e Rotina OPERA, respectivamente. As rotinas são programadas pelo usuário na linguagem Pascal Script, que será abordada com mais profundidade em tópicos posteriores.

A rotina PLANEJA elaborada pelo usuário é executada no início de cada intervalo de tempo, impondo uma estratégia operacional. A rotina OPERA, por sua vez, é executada durante o método Balanço Hídrico de cada ponto característico que possua um reservatório.

2.4.4 A Simulação da Geração de Energia Hidroelétrica no Propagar MOO

O modelo permite simular em cada PC a existência de uma usina hidrelétrica. A geração de energia é computada por um método denominado *CalculaEnergia*, que possui uma função padrão para o cálculo da energia gerada em cada PC. Este cálculo é realizado no final do balanço hídrico de cada PC que tenha a geração de energia habilitada.

Conforme será abordado no Capítulo 3 desse trabalho, uma nova função padrão de cálculo de energia em um PC foi implementada no corpo do modelo. Ela utiliza os dados de rendimento do sistema de adução, rendimento das turbinas, rendimento dos geradores, vazão máxima turbinável e cota da usina, todos fornecidos individualmente pelo usuário, além dos dados de vazão e nível calculados ao longo da simulação.

Porém, o usuário pode implementar um cálculo diferente do padrão através de uma rotina denominada Cálculo de Energia, programada na linguagem Pascal Script. Essa rotina é executada após o cálculo de energia padrão.

2.4.5 Ferramentas para Avaliação dos Resultados de Simulações

Como o propósito do modelo Propagar MOO é realizar a simulação da propagação de vazões, visando o atendimento de demandas em função da disponibilidade de água, a avaliação do atendimento dessas demandas e das regras operacionais utilizadas para tal deve ser o objetivo maior da análise dos resultados (Viegas Fo, 2000).

As falhas de atendimento às demandas podem ser analisadas de várias formas, como, por exemplo, verificado o número de intervalos de tempo onde determinada demanda não é suprida ou sendo seu suprimento inferior a um determinado valor considerado crítico.

O Propagar MOO possui algumas ferramentas nativas de análise dos resultados da simulação, disponíveis através dos menus do modelo. Dentre elas pode-se citar o Relatório Geral de Falhas, no atendimento das demandas, bem como a possibilidade de acesso, através de planilhas e gráficos, às propriedades de cada PC ao final da simulação – inclusive para a análise individual das falhas em Demandas Primárias, Secundárias e Terciárias ocorrida em cada PC.

Quanto às propriedades dos PCs em cada intervalo de tempo, acessíveis ao final da simulação, pode-se citar as afluências e defluências de cada PC, as vazões de montante, as demandas atendidas, planejadas e totais, a energia gerada em um PC, ou, ainda, os volumes dos reservatórios ao longo da simulação.

A análise dos resultados da simulação também pode ser realizada através de funções programadas pelo usuário através de “scripts”. Para tal existe uma rotina, denominada Rotina de Uso Geral, executada pelo modelo ao final da simulação após o último intervalo de tempo de simulação.

A Rotina de Uso Geral permite ao usuário a implementação de algorítimos para a análise de resultados, sejam de cálculo ou estatísticos, ou, ainda, algorítimos para a construção de gráficos e tabelas que auxiliem na análise dos resultados. A rotina permite, também, a inclusão de funções de rastreamento (debug), possibilitando ao usuário, a programação de quaisquer testes durante a simulação.

2.5 A LINGUAGEM PASCAL SCRIPT COMO FERRAMENTA DE PLANEJAMENTO E ANÁLISE NO PROPAGAR MOO

2.5.1 Fundamentos Básicos da linguagem PASCAL SCRIPT

A linguagem de programação Pascal Script é uma linguagem que utiliza uma estrutura sintática e léxica semelhante à da a linguagem de programação Pascal. Foi construída e integrada ao Propagar MOO (Viegas F^{o}, 2000) a partir de um trabalho acadêmico desenvolvido por Conceição (2000), sobre a construção de um compilador baseado em Pascal.

Segundo Viegas F^{o} (2000), a idéia de criar um compilador baseado na linguagem Pascal surgiu da necessidade de atrair o interesse dos usuários já acostumados com o modelo Propagar em sua versão DOS, para o Propagar MOO. Ocorre que tais usuários, normalmente engenheiros familiarizados com linguagens procedurais como o FORTRAN, já utilizavam as vantagens de poder programar as rotinas Planeja e Opera e, provavelmente, teriam alguma dificuldade em vencer a barreira da programação em Object Pascal e em dominar o ambiente de desenvolvimento Delphi.

Além disso, a criação do compilador também foi impulsionada pela decisão de não manter o código do PROPAGAR MOO totalmente aberto, pois o usuário, mesmo sabendo programar, teria acesso a todo o modelo. Isso permitiria alterações em rotinas que dão consistência à Rede Hidrológica e ao mecanismo lógico de simulação. O uso incorreto desse recurso poderia comprometer o funcionamento do modelo (Viegas F^{o},2000).

Dessa forma, foi implementada a Linguagem Pascal Script, cuja estrutura básica será mostrada a seguir (adaptado de Viegas F^{o}, 2000).

A programação de rotinas em Pascal Script pode ser feita em qualquer editor de textos, entretanto, para tornar isso mais fácil ao usuário, o PROPAGAR MOO coloca a sua disposição um editor especialmente construído para tal fim, denominado Editor Pascal Script (Conceição, 2000). O editor é ativado a partir do menu do Propagar MOO.

A Figura 2.5 apresenta o ambiente de desenvolvimento do Editor de Script do Propagar MOO, indicando sua áreas internas.

O ambiente de desenvolvimento desse editor de Scripts possui uma área de funções que relaciona todas as classes de objetos, métodos de classes (funções e

procedimentos aplicáveis a cada classe de objeto) e rotinas de uso geral (funções e procedimentos gerais) que estão disponíveis para o usuário através das bibliotecas associadas ao compilador.

Na medida em que uma função é selecionada na área de funções, aparece automaticamente, na barra de comentários ao pé do editor, a descrição da mesma e dos parâmetros requeridos, quando for o caso. Ao dar um duplo-clique com o ponteiro do mouse sobre qualquer função, esta imediatamente será inserida na área de edição, no local onde se encontrar o cursor de edição.

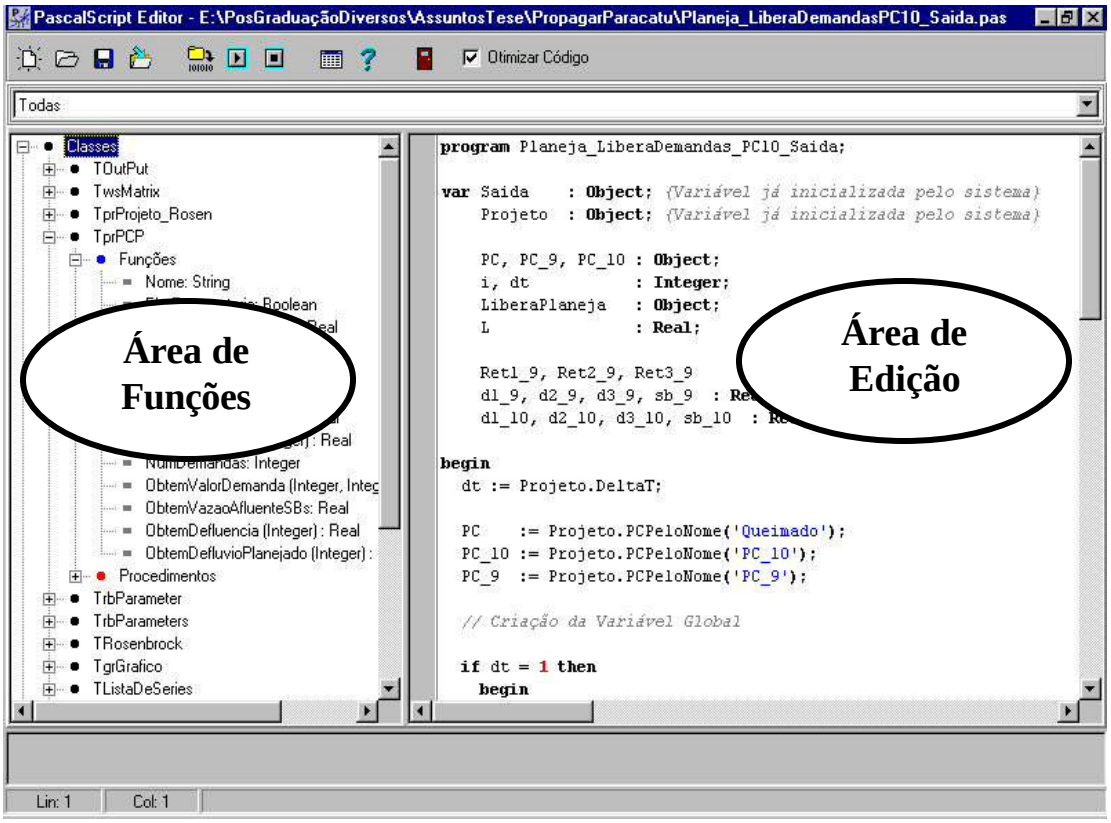


Figura 2.5 – Ambiente de desenvolvimento do Editor Pascal Script.

As rotinas em Pascal Script são construídas exatamente nessa área de edição, que possui recursos para tal, como o que destaca com cores distintas as diferentes partes do código (palavras reservadas, comentários e strings, número, etc.), o que facilita muito a edição e legibilidade do código.

Outro recurso do editor é o de correção da rotina escrita pelo usuário, durante a compilação do código, onde o editor informará ao usuário a existência de erros e o local onde se encontram.

De forma simplificada, um programa em Pascal Script constitui-se de uma seqüência de unidades de texto, chamadas “Tokens”, delimitadas por separadores, como espaços em branco ou comentários. Essas unidades de texto são classificadas em categorias, como símbolos especiais, identificadores, palavras reservadas, números e caracteres strings. Identificadores adjacentes, palavras reservadas e números devem possuir pelo menos um separador entre eles.

Um identificador é uma unidade de texto que tem por objetivo fornecer um nome capaz de identificar uma variável, função ou procedimento e sua formação deve seguir as seguintes regras: o primeiro caractere do identificador deverá ser uma letra ou um símbolo de sublinhado; os demais caracteres podem ser letras, dígitos ou sublinhados; não pode ser uma palavra reservada do Pascal Script.

As palavras reservadas são palavras que fazem parte da estrutura do Pascal Script e que possuem significados pré-determinados. Não podem ser, portanto, redefinidas e nem utilizadas como identificadores de variáveis, procedimentos e funções.

As palavras reservadas do Pascal Script são: *program, var, string, integer, real, boolean, object, begin, end, while, do, for, to, downto, if, then, else, not, and e or*.

Strings são conjuntos de caracteres que, no Pascal Script, devem ser escritos entre aspas simples, como por exemplo, ‘`texto qualquer`’.

Os comentários constituem-se por textos introduzidos no meio do código do programa com a intenção de torná-lo mais claro. Em Pascal Script, tudo que estiver entre os símbolos "{" e "}" ou ainda, após o símbolo "//", porém, em uma mesma linha, será considerado como comentário. A seguir apresenta-se um exemplo da sintaxe dos comentários.

```
// Isto é um comentário de uma linha

{ Isto é um bloco
com duas linhas de comentário}
```

Um programa em Pascal Script consiste inteiramente de declarações e comandos, os quais são organizados em três blocos: Cabeçalho do programa (opcional), Bloco de declarações de variáveis (área de declaração) e Bloco principal (corpo do programa), conforme o exemplo abaixo.

```
program Identificacao; // cabeçalho

    var                     // início da área de declaração
       v1, v2: Tipo;        // declaração de variáveis
       v3, v4: Tipo;

    begin                   // início do corpo do programa - bloco principal
       comando 1;           // composto por comandos simples ou compostos
       comando 2;
       ...
       comando N;
    end.
```

Denomina-se variável um valor que pode ser alterado dentro de um programa. O bloco de declarações é o local onde são definidas as variáveis que não tenham sido previamente declaradas na própria linguagem. No caso de serem utilizadas somente variáveis pré-declaradas, como por exemplo as variáveis *saida e projeto*, utilizadas em projetos do Propagar MOO, esta área pode ser suprimida. Entretanto, quando o usuário desejar definir novas variáveis dentro do "script" deverá, obrigatoriamente, fazer a sua declaração.

O PascalScript permite a declaração de 5 tipos de variáveis: *Integer, Real, Boolean, String e Object*. Todas as variáveis utilizadas no programa devem ser declaradas na área de declaração, usando a cláusula *Var*, para que a alocação de espaço de memória para as mesmas seja feita durante a compilação.

Os comandos são unidades de texto que definem as ações dentro do algoritmo em um programa. Existem comandos simples como, por exemplo, um comando de atribuição ou chamadas de procedimentos e comandos estruturados, como comandos que formam laços, comandos condicionais, entre outros.

O comando de atribuição é usado para atribuir o valor de uma expressão a uma variável. Sua sintaxe é:

```
Variável := Expressão;
```

Comandos compostos consistem de um ou vários comandos simples executados seqüencialmente e que são usados em situações onde a sintaxe de alguns comandos do Pascal permite apenas um único comando.

Os comandos que fazem parte de um comando composto são separados por ponto-e-vírgula, devendo começar e terminar, respectivamente, pelas palavras reservadas BEGIN e END, de acordo com a sintaxe demonstrada a seguir:

```
begin
comando [ ; comando ]...
comando [ ; comando ]...
end;
```

Os comandos estruturados são construídos com base em outros comandos. São utilizados quando se deseja a execução de outros comandos de forma seqüencial, condicional ou repetida. Um comando estruturado pode ser formado por um ou mais blocos constituintes.

O comando condicional IF, que executa um dos seus dois blocos constituintes dependendo do critério especificado, é um exemplo de comando estruturado. Outro exemplo pode ser dado por comandos que formam laços, tais como os comandos WHILE e FOR, e que executam uma seqüência de comandos repetidamente.

As expressões do Pascal Script são constituídas de conjuntos de operandos unidos por operadores, de forma a computar um valor ou resultado. Os operandos podem ser variáveis, constantes ou valores gerados por funções, enquanto que os operadores identificam as operações a serem efetuadas sobre os operandos. Existem diferentes tipos de operadores.

Os operadores aritméticos, comuns à diversas linguagens, são: + (adição), - (subtração), * (multiplicação), / (divisão real), *div* (divisão inteira) e *mod* (resto de uma divisão inteira).

Existem também os operadores lógicos ou Booleanos: *AND* (E lógico) e *OR* (OU lógico). Esses operadores só aceitam como operandos, valores lógicos, ou seja, valores *TRUE* ou *FALSE*. A operação *AND* resulta em *TRUE* se e somente se todos os operandos forem *TRUE*; caso um deles ou mais de um for *FALSE* então o resultado será *FALSE*. A operação *OR* resulta *TRUE* quando pelo menos um dos operandos for *TRUE*.

Existem ainda os operadores relacionais do Pascal Script. São 6 operadores muito utilizados nas tomadas de decisões e no controle de fluxo, são eles: = (igual), <> (diferente), > (maior que), < (menor que), >= (maior ou igual que) e <= (menor ou igual que).

O Pascal Script possui funções e procedimentos pré-definidos, cuja chamada consiste em uma referência ao nome dessa função ou procedimento, seguido de sua lista de parâmetros, quando requerido. Uma função ou um procedimento pode pertencer a uma classe de objetos e, nesse caso, faz parte dos métodos dessa classe. A única diferença entre funções e procedimentos é que a primeira retorna um valor que pode ser atribuído a uma variável ou utilizado em uma expressão.

Dentro de um código em Pascal Script pode surgir a necessidade de tratar um determinado objeto como se fosse de outro tipo. Para isso existe um procedimento de programação chamado “typecast”, que, em outras palavras, serve para converter um tipo em outro. A sintaxe para o typecast é:

```
tipoIdentificador(variável Object);
```

Para permitir a tomada de decisões e a execução de múltiplas repetições de um mesmo comando ou bloco de comandos existem os comandos de controle de fluxo.

O primeiro deles é o comando condicional *IF*, já mencionado, que permite ao programa tomar decisões e pode ter duas sintaxes diferentes. Na primeira, demonstrada abaixo, se a *expressão_lógica* resultar em um valor verdadeiro (*TRUE*), então o comando será executado, caso contrário, não. Nos caso em que tenham de ser executados mais de um comando eles deverão ser acomodados em um bloco "*Begin ... End*", já abordado anteriormente.

```
if expressão_lógica then Comando;
```

Na segunda sintaxe, apresentada abaixo, se *expressão_lógica* for *TRUE* então *comando_1* será executado e *comando_2* não, caso contrário, *comando_2* será executado e *comando_1* não. Repare-se que não foi colocado ";" no final de *comando_1* uma vez que, nesse momento, o comando ainda não está terminado.

```
if expressão_lógica then
       Comando_1
    else
       Comando_2;
```

Um segundo comando de controle de fluxo é o comando *FOR*, que permite ao programador executar uma seqüência de comandos de modo repetido, usando uma condição de controle ou variável para determinar quando a execução deverá ser terminada.

Uma demonstração da sua sintaxe é apresentada abaixo.

```
for [variável] := [valor_inicial] to/downto [valor_final]
[step incremento] do comando;
```

A variável deverá ser do tipo Integer ou Real. O comando faz com que a variável altere seu valor entre valor inicial e valor final, de "*step*" em "*step*". A alteração será de forma crescente quando se utiliza a palavra *to* e de forma decrescente quando é utiliza-se a palavra *downto.* Se o parâmetro *step* for omitido, o valor assumido para o incremento/decremento da variável será 1.

Em Pascal Script, além do comando de laço *FOR*, existe o comando *While...Do*. A sua estrutura permite que um comando ou um bloco de comandos seja executado enquanto o valor ou argumento contido na *expressão lógica* seja verdadeiro. Sua sintaxe básica é a seguinte:

```
while expressão_logica do comando;
```

A Pascal Script é uma linguagem baseada em objetos e, como tal, permite a criação de instâncias de classes. A maior parte dos objetos podem ser criados através do comando *CreateObject()*. Sua sintaxe é demonstrada abaixo.

```
CreateObject('Nome da Classe')
```

Além de todos os comandos apresentados, a Pascal Script aplicada ao Propagar MOO possui outros comandos e uma grande lista de bibliotecas de funções e procedimentos que podem ser utilizados pelos programadores. Muitos desses recursos foram utilizados pelo autor dessa dissertação e, no devido momento, serão abordados com mais detalhes.

Para maiores esclarecimentos sobre a linguagem de programação Pascal Script, indica-se Viegas F^{o}(2000), que possui um manual completo da linguagem.

2.5.2 O PASCAL SCRIPT aplicado ao PROPAGAR

O desenvolvimento inicial do Pascal Script aplicado ao Propagar deu-se em torno da possibilidade de acesso aos atributos e métodos dos objetos de uma Rede Hidrográfica construída no Propagar MOO, permitindo ao usuário a programação das rotinas Planeja e Opera. (Viegas F^{o},2000).

Porém, atualmente, o Pascal Script pode ser utilizado para implementação de outras rotinas dentro do modelo, programáveis pelo usuário. São elas: Planeja, Opera, Cálculo de Energia, Rotina de Uso Geral e, ainda, as rotinas denominadas Scripts Gerais, que podem ser executadas de forma independente da simulação da rede hidrográfica.

A rotina Planeja é executada no início de cada intervalo de tempo, antes de ser iniciada a propagação da água através da Rede. Dessa forma, o usuário pode simular todo um processo de tomada de decisões, na forma de planejamento do uso da água, no início de um intervalo de tempo. (Viegas F^{o},2000).

Uma das principais aplicações dessa rotina é o planejamento da distribuição da água, em função da disponibilidade existente nos reservatórios da bacia, no início de cada intervalo de tempo. Entretanto, pode-se programar, também, tomadas de decisões para longo prazo.

A rotina Opera, por sua vez, é executada em cada reservatório existente dentro da Rede Hidrográfica, sendo chamada a cada intervalo de tempo, através do método Balanço Hídrico de um reservatório, após a verificação da sua disponibilidade de água. A rotina pode ser usada para testar diferentes regras de operação em função da quantidade de água que, no transcorrer do intervalo de tempo, de fato, estará presente no reservatório (Viegas F^{o},2000).

Além da rotina Opera geral, que será executada para todos os reservatórios existentes na rede hidrográfica, o usuário pode criar uma rotina Opera específica para cada um dos reservatórios existentes na rede. Uma vez existindo a rotina Opera específica, esta será executada, em detrimento da rotina Opera geral.

A rotina Cálculo de Energia possibilita ao usuário simular a geração de energia na rede hidrográfica, através da programação de funções matemáticas para o cálculo da energia gerada nos PCs do sistema que tiverem essa destinação. A rotina é chamada no final do balanço hídrico de um PC (Viegas F^{o},2000). De forma similar à rotina Opera, o usuário

também pode criar rotinas de Cálculo de Energia específicas para cada PC que possuir usina para geração de energia, além da rotina de Cálculo de Energia genérica a ser executada em todos os PCs com geração de energia ativada. A primeira, se existir, será executada no lugar da segunda.

A Rotina de Uso Geral é executada no final de toda a simulação, possibilitando ao usuário, inclusive, a realização de operações de rastreamento e cálculos finais, bem como a apresentação de resultados em planilhas, na forma de gráficos e/ou no editor de textos do Propagar MOO, desde que esses estejam disponíveis na forma de atributos de objetos e/ou em variáveis ou objetos globais que tenham sido criados com essa finalidade (Viegas F^{o},2000).

Os Scripts Gerais podem ser executados em qualquer momento, desde que a simulação não esteja em andamento. Através deles, por exemplo, o usuário pode acessar e analisar qualquer conjunto de dados, localizados em arquivos ou em propriedades dos objetos da rede, utilizando ferramentas de cálculo ou estatísticas e demonstrar os resultados através de gráficos, tabelas e/ou no editor de textos do Propagar MOO.

A Figura 2.6 apresenta os níveis de abrangência das chamadas de rotinas criadas pelo usuário dentro do modelo Propagar MOO, citando, para cada uma, um exemplo dentre as rotina implementada nesse trabalho.

Os Scripts Gerais podem acessar quaisquer propriedades e métodos dos objetos que integram o projeto em estudo, quando estes estiverem disponíveis como funções do Pascal Script. São executados fora de um processo de simulação.

A rotina Planeja é chamada dentro da simulação, no início de cada intervalo de simulação, sendo única para toda a rede hidrográfica.

A rotina Opera é chamada dentro do nível de propagação de água na rede, no início do balanço hídrico de um PC com reservatório. Pode existir uma rotina Opera válida para todos os reservatórios da rede e/ou rotinas de operação específicas para cada um deles.

A rotina de Cálculo de Energia é chamada ao final do balanço hídrico de cada PC, ainda dentro do nível de propagação de água na rede hidrográfica. Pode existir uma rotina de Cálculo de Energia válida para todos os PCs com geração de energia da rede e/ou rotinas de Cálculo de Energia específicas para cada PC com sistema de geração de energia.

A Rotina de Uso Geral é chamada ao final da simulação da rede hidrográfica.

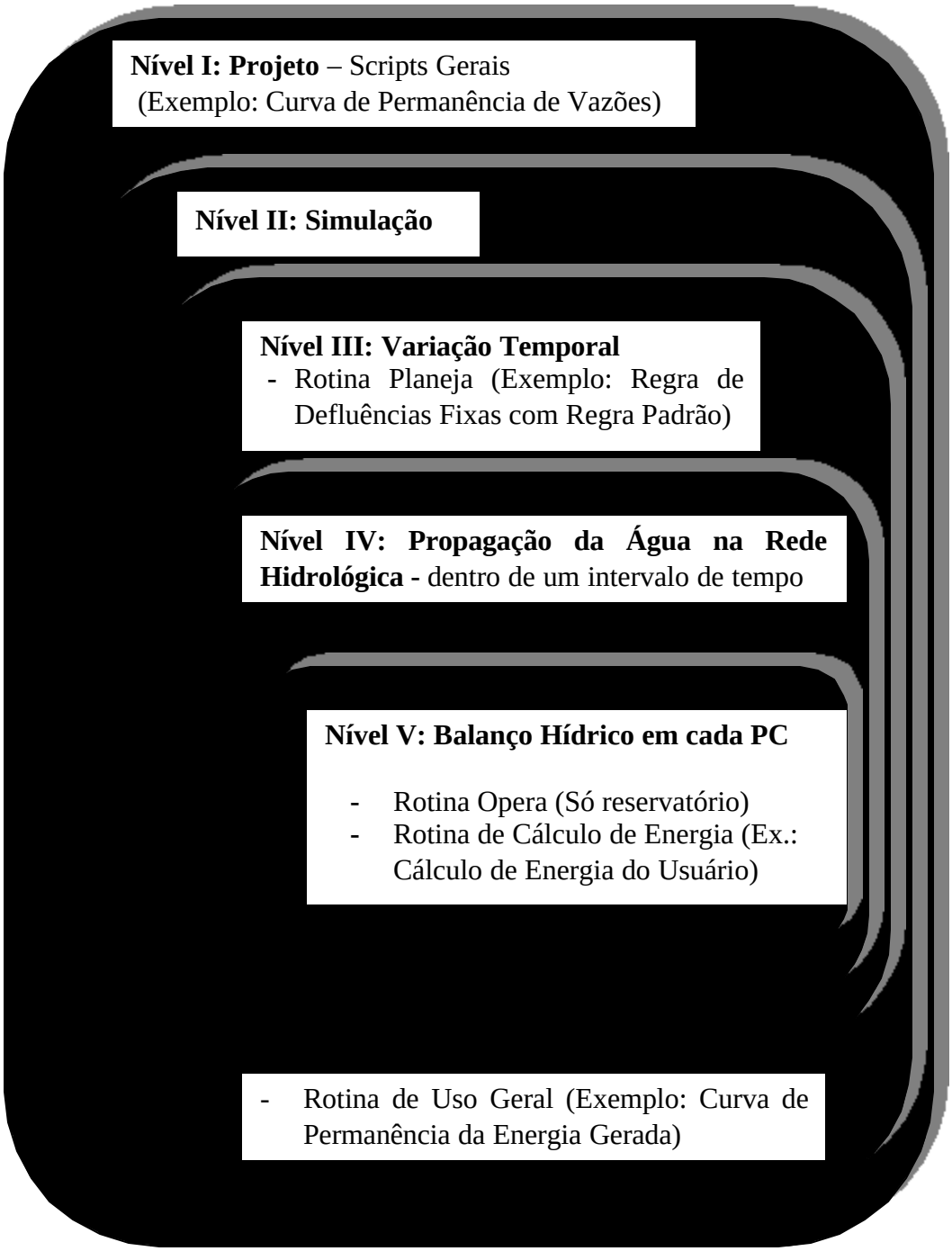


Figura 2.6 - Níveis de abrangência das rotinas PASCAL SCRIPT no PROPAGAR MOO. (Adaptado de Viegas F^{o}, 2001).

Observa-se que a possibilidade de execução de rotinas programadas pelo usuário torna o modelo Propagar MOO extremamente flexível. Além disso, a utilização de “scripts” e de um compilador Pascal possibilita o desenvolvimento de ferramentas de análise de resultados, dentro de um amplo espectro: econômicas, estatísticas, de análise de risco, dentre outras. (Viegas F^{o},2000).

CAPÍTULO 3 - METODOLOGIA.

3.1 CONSIDERAÇÕES PRELIMINARES.

No capítulo anterior foi demonstrada a importância em tratar de forma sistêmica os projetos da área de recursos hídricos, bem como a gestão de qualquer aproveitamento.

Observou-se que uma das metodologias mais utilizadas para esse fim é a simulação aplicada a sistemas de recursos hídricos, através da qual já foram desenvolvidos vários modelos para representar os mais diversos sistemas. Um desses modelos é o PROPAGAR, cuja concepção e dinâmica básica de simulação já foram abordadas.

O modelo PROPAGAR permite ao seu usuário a possibilidade de testar diferentes regras operacionais em sistemas de recursos hídricos, através de rotinas elaboradas pelo próprio usuário. O modelo também permite simular a geração de energia hidroelétrica em um sistema hídrico por ele representado, além de possuir algumas ferramentas nativas para facilitar a análise dos resultados da simulação.

Este capítulo apresenta, inicialmente, os métodos desenvolvidos na linguagem Pascal Script, visando instrumentalizar o modelo Propagar MOO com ferramentas genéricas de planejamento do uso da água e para operação de reservatórios.

Na seqüência, são apresentadas novas ferramentas para simular a geração de energia hidrelétrica em pontos de uma rede hidrográfica.

Por fim, são apresentadas maneiras de como o próprio usuário pode construir ferramentas para a análise dos resultados de simulações, bem como utilizar os Scripts Gerais, executados independentemente da simulação da rede, como ferramentas de análise de dados.

3.2 FERRAMENTAS DE PLANEJAMENTO DO USO DA ÁGUA E OPERAÇÃO DE RESERVATÓRIO.

Conforme já abordado anteriormente, o modelo Propagar permite que o usuário desenvolva regras operacionais para planejar o uso da água e operar reservatórios durante a simulação de um sistema hídrico. Isso é feito através de rotinas de planejamento que o usuário pode implementar, utilizando a linguagem de programação Pascal Script.

Existem usuários tradicionais do modelo Propagar que desejam, ou necessitam, criar e implementar estratégias operacionais em suas aplicações, porém, o estudo de uma linguagem de programação exige certa quantidade de tempo e dedicação que nem sempre os usuários dispõem.

Para esses casos seria interessante a apresentação de rotinas de planejamento genéricas, ou seja, que pudessem ser aplicadas em diferentes estudos ou bacias hidrográficas, sem a necessidade de alterações significativas, bem como demonstrar a implementação dessas estratégias operacionais e a análise de seus resultados, preferencialmente através de exemplos.

Para atender a essa demanda, surgiu, então, a idéia de desenvolver ferramentas genéricas de planejamento do uso da água e de operação de reservatórios em sistemas de recursos hídricos, que poderiam ser aplicadas em diferentes casos, efetuando-se pouca ou nenhuma modificação nos seus procedimentos e funções.

Para tal, foram selecionadas algumas estratégias operacionais, já abordadas conceitualmente, para serem implementadas como ferramentas genéricas de planejamento do uso da água e operação de reservatório.

São elas: Defluências Fixas com Regra Padrão de Decisão, Regra Padrão Modificada com Procedimento para a Avaliação de Demandas, Regra dos Volumes-Metas com Procedimento para a Avaliação de Demandas e Regra de Zoneamento de Reservatório com Procedimento para a Avaliação de Demandas.

3.2.1 Defluências Fixas com Regra Padrão de Decisão

A regra de operação que estabelece uma defluência fixa, utilizada em conjunto com a chamada regra padrão de decisão - para o caso do não atendimento completo das demandas - foi selecionada para ser a primeira regra operacional genérica implementada no modelo Propagar MOO devido à sua simplicidade.

Basicamente essa regra permite que o usuário informe quais as defluências que deseja liberar do reservatório para cada período do ano, informando também, em que ponto da rede encontra-se o reservatório a ser operado segundo a regra.

No caso de existir, no período em simulação, uma disponibilidade hídrica que possa suprir a defluência solicitada, esta será atendida. Caso contrário, será aplicada a Regra Padrão de Decisão, que, conforme foi visto anteriormente, caracteriza-se por determinar o atendimento das demandas enquanto houver água disponível, podendo, inclusive, liberar toda a água disponível, no caso das demandas previstas superarem a disponibilidade hídrica, provocando um atendimento parcial das demandas. Por causa disso, a Regra Padrão de Decisão é considerada uma regra de operação “míope”, pois utiliza a água disponível até o seu limite, sem preservá-la para possíveis utilizações futuras.

Para melhor compreensão da aplicação da regra padrão de decisão, dentro do modelo Propagar MOO, é importante uma descrição da estrutura interna do modelo, no que se refere às etapas de planejamento e de operação.

Ao executar a etapa de planejamento, primeiramente o modelo atribui para as demandas planejadas de todos os PCs os valores das respectivas demandas totais, ou seja, faz com que se tente atender todas as demandas previstas para cada PC. Ao avaliar cada PC, o modelo também verifica se o PC possui reservatório, o que, em caso afirmativo, provoca a definição de uma defluência planejada igual a zero, para esse reservatório.

Em um segundo momento, o modelo executa a rotina de planejamento do usuário, se esta existir. Nesse ponto poderá ser planejada uma defluência diferente de zero, o que é feito na rotina de defluências fixas, descrita nesse item.

Após isso, na etapa de balanço hídrico de cada PC, é aplicada a regra padrão de decisão incorporada à estrutura do modelo, onde são atendidas as demandas conforme a disponibilidade de água. No caso de ser um reservatório, o modelo, através da regra padrão de decisão, busca atender as demandas do próprio PC e, também, a defluência planejada anteriormente, desde que exista água disponível no reservatório.

Dessa forma, o racionamento é executado normalmente pelo próprio modelo, sem a necessidade de sua implementação dentro da rotina de planejamento, desde que o usuário faça a definição das defluências na rotina de planejamento.

No caso de haver excesso hídrico no período, a regra estabelece um vertimento que, no mínimo, atende a defluência estabelecida pelo usuário para o período em simulação.

Tanto o racionamento no atendimento das defluências fixadas pelo usuário em cada período, quando não há disponibilidade, como o vertimento, quando há excesso hídrico, são ilustrados pela Figura 3.1.

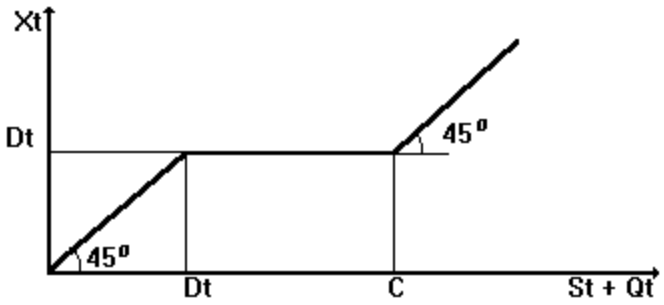


Figura 3.1: Regra Padrão de Decisão,
(Fonte: Vianna Jr., 1998).

Na Figura 3.1, o eixo das ordenadas representa a descarga $\mathbf{X_t}$ do reservatório em um intervalo de tempo **t** qualquer. O eixo das abcissas representa a disponibilidade hídrica no intervalo de tempo, onde $\mathbf{S_t}$ é o volume de água armazenado no reservatório no início do intervalo de tempo **t** e $\mathbf{Q_t}$ é a contribuição hídrica ao reservatório (já descontadas as perdas durante o mesmo intervalo de tempo). O termo $\mathbf{D_t}$ representa a defluência solicitada pelo usuário, ou seja, a demanda a ser suprida pelo reservatório durante o intervalo de tempo **t**.

Nessa regra operacional, sempre que a disponibilidade hídrica for inferior à defluência a ser atendida, a decisão adotada é liberar toda a água disponível. Essa decisão é representada pelo primeiro trecho da função, segundo uma reta a 45^o, onde a descarga é igual à disponibilidade hídrica.

Quando a disponibilidade hídrica for igual ou superior à defluência solicitada, esta será atendida integralmente, conforme representado no trecho da Figura 3.1, onde $\mathbf{X_t = D_t}$.

No caso do reservatório atingir a sua capacidade máxima (**C)** ou um volume máximo definido pelo usuário (através da definição de um volume de espera para amortecer cheias), a demanda também será integralmente atendida, sendo que a disponibilidade hídrica excedente será extravasada pelo reservatório. Essa operação é ilustrada pela reta com inclinação de 45^o, no trecho final da curva.

Observa-se que essa regra não realiza, por si só, a avaliação das demandas hídricas à jusante do reservatório. A regra pressupõe que a avaliação de demandas a serem atendidas seja feita pelo próprio usuário do modelo e o resultado da avaliação introduzido no mesmo, na forma de defluências planejadas para cada período do ano.

Para tornar essa regra genérica, ao ser implementada no modelo, definiu-se que a própria rotina de planejamento deveria buscar, fora do ambiente do programa, os valores de defluência planejada e os valores dos volumes de espera desejados, para cada mês do ano. Esses dados seriam armazenados em um arquivo padrão, do tipo “planilha”, construído pelo usuário e descrito posteriormente.

A Figura 3.2 apresenta o fluxograma básico da rotina de planejamento que implementa no modelo, a regra operacional descrita acima.

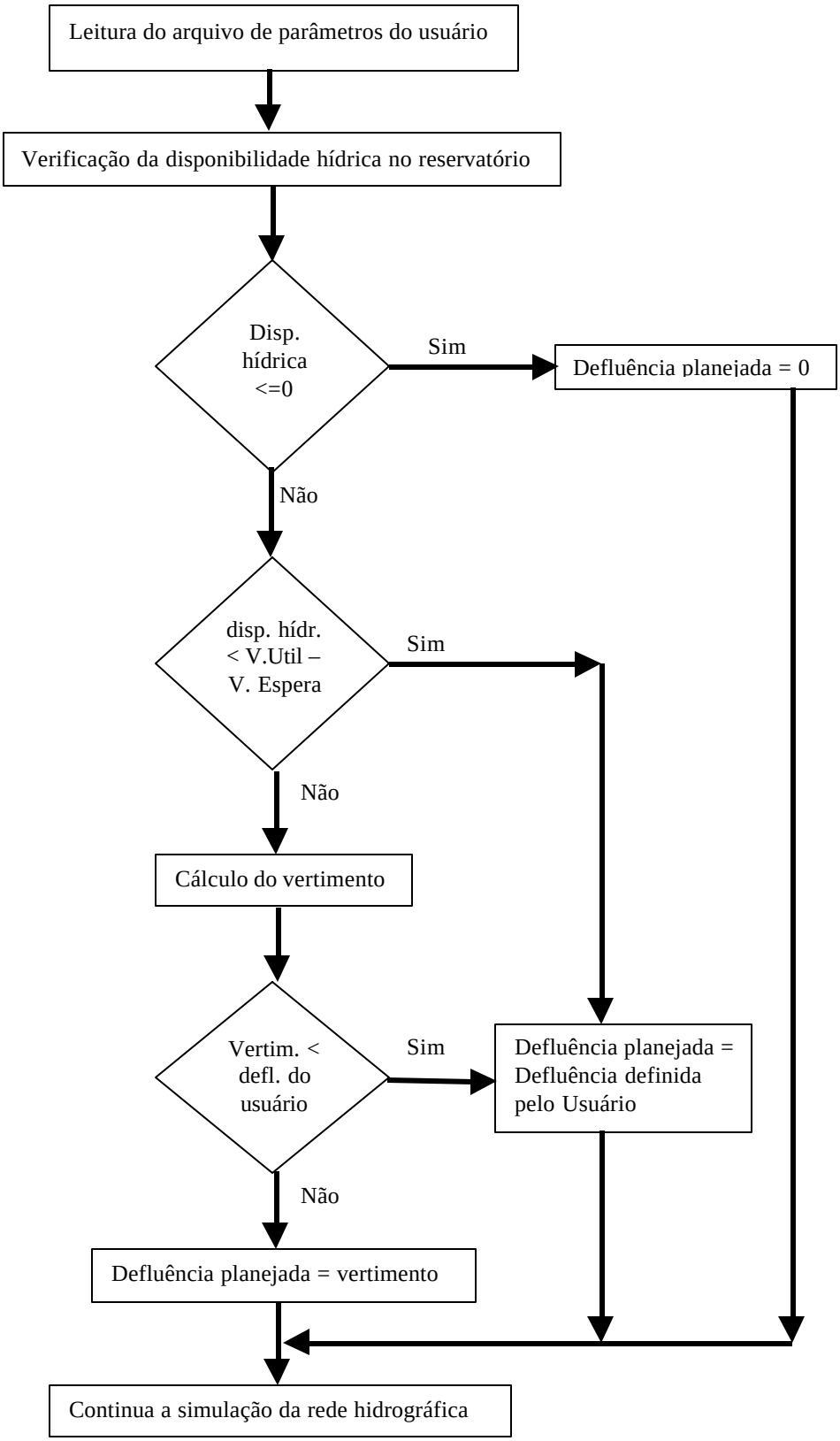


Figura 3.2: Fluxograma básico da regra de defluências fixas.

A seguir, serão apresentados trechos das rotinas construídas na linguagem Pascal Script, acompanhados de comentários e esclarecimentos.

A etapa de entrada de dados do usuário é apresentada abaixo, em Pascal Script.

```
...

Plan := CreateObject(TPlanilha);
Plan.LoadFromFile('D:\Diretorio\Arquivo.xls'); // Faz a leitura do arquivo
PCReserv := Plan.GetEntry(1,1); // Copia dado da planilha p/ variável
vDefluFixo := Plan.RowToVec(3,1,12); // Copia dados da planilha p/ vetor
vVEspera := Plan.RowToVec(5,1,12);

...
```

Primeiramente é criado um objeto do tipo planilha, identificado pelo nome *Plan*. Na verdade, *Plan* é uma variável do tipo objeto, que deve ser primeiramente declarada na seção de declaração de variáveis, após a cláusula *Var*, como as demais variáveis usadas nas rotinas escritas em Pascal Script.

Na linha seguinte, utiliza-se o procedimento *LoadFromFile*, que realiza a leitura de um arquivo padrão Excel 4.0 e transfere seu conteúdo para o objeto “planilha” criado. Salienta-se que o nome e a localização do arquivo criado pelo usuário devem ser iguais aqueles estipulados na chamada do arquivo, no interior da rotina de planejamento, como por exemplo ‘D:\Diretorio\Arquivo.xls’, utilizado no trecho de rotina apresentado acima.

As três linhas finais do trecho referem-se à transferência dos dados fornecidos pelo usuário, do objeto planilha para os vetores e variáveis que serão utilizados ao longo da rotina. Observa-se que a transferência ocorre a partir de posições pré-definidas do objeto planilha e que são iguais às posições dos dados dentro do arquivo de entrada.

O arquivo de entrada de dados criado pelo usuário deve ser gravado no padrão Excel 4.0 em função das bibliotecas de planilhas utilizadas no ambiente Delphi para o desenvolvimento do modelo Propagar MOO apresentarem essa limitação.

Para que a rotina seja executada corretamente, o arquivo deve possuir os seguintes dados: a identificação do PC (dentro da rede) onde está o reservatório, os 12 valores das defluências a serem estabelecidas no reservatório em cada mês do ano (em m^3/s) e os 12 valores definidos como volumes de espera para o reservatório, estes indicados em percentual do volume máximo do reservatório. Salienta-se a importância da observação da organização interna dos dados no arquivo, apresentada na Tabela 3.1, para que não ocorram erros durante a execução do “Script”.

Tabela 3.1 – Exemplo de formato do arquivo de entrada de dados para a rotina de planejamento que aplica a regra de Defluências Fixas.

PC reserv.											
Defluência	Fixa	(m^3/s)									
40	35	30	25	20	15	15	20	25	30	35	40
Volumes de	Espera	(% Vol.	Max)								
10	9	8	7	6	5	5	6	7	8	9	10

O trecho da rotina que realiza o cálculo da disponibilidade hídrica do reservatório, em cada intervalo de tempo, é apresentado a seguir:

```
...

VolDisp := VolInicInterv - PCR.VolumeMinimo;
AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
VazaoPCsMontante := 0;
if PCR.PCs_aMontante > 0 then
  begin
  for II := 0 to (PCR.PCs_aMontante - 1) do
    begin
    PCMontante := PCR.PC_aMontante(II);
    Projeto.RealizaBalancoHidricoAte(PCMontante);
    VazaoPCsMontante := VazaoPCsMontante + PCMontante.ObtemDefluencia(dt);
    end;
  VazaoPCsMontante := PCR.m3_Hm3_Intervalo(VazaoPCsMontante);
  end;
Disp := VolDisp + AfluHm3 + VazaoPCsMontante;

...
```

Primeiramente avalia-se o volume de água disponível no reservatório, no início do intervalo de tempo, e as vazões provenientes de sub-bacias a montante que contribuem para o reservatório. *PCR.VolumeMinimo* e *PCR.ObtemVazaoAfluenteSBs* exemplificam como métodos de um PC podem ser utilizados para fornecer um valor armazenado em um atributo do mesmo; no caso, o volume mínimo do reservatório e as vazões afluentes de sub-bacias que contribuem para o mesmo.

Após isto, verifica-se a presença de PCs a montante do reservatório e, no caso de existirem, aplica-se um conjunto de instruções em laço ou “Loop” para contabilizar a defluência de cada PC encontrado a montante do reservatório. *PCR.PCs_aMontante* é um método da classe Projeto que dá acesso à lista de PCs; no caso, *PCs_aMontante* fornece o número de PCs localizados à montante do PC em estudo.

Porém, para avaliar corretamente a defluência de um PC que contribua para o reservatório, é necessário que se verifique toda a rede de PCs e sub-bacias que estiver a montante desse PC, contemplando as afluências hídricas e o atendimento das demandas de cada um. Isso significa realizar um balanço hídrico até o PC em questão.

Para isso foi criado um novo procedimento que realiza o balanço hídrico até um determinado ponto da rede hidrográfica, ou seja, um balanço hídrico parcial da rede. O novo procedimento é denominado *RealizaBalancoHidricoAte("nome do PC")*, sendo esta uma das ferramentas desenvolvidas especialmente para o presente trabalho e que acrescentam recursos à linguagem Pascal Script do Propagar MOO. Esse procedimento é abordado em detalhes no Anexo IV do presente trabalho.

Nessa avaliação da disponibilidade hídrica do reservatório não foram consideradas a precipitação e a evaporação sobre a superfície do mesmo. Isso foi adotado para manter a simplicidade da rotina, pois a avaliação da influência dessas variáveis no processo depende da estimativa da superfície livre do reservatório, o que implicaria na construção de uma rotina com iterações de cálculo para contemplar a variação da área e do volume do reservatório no intervalo de tempo.

No final dessa etapa, realiza-se o cálculo de totalização da disponibilidade hídrica do PC reservatório, no intervalo de tempo considerado.

O trecho da rotina de planejamento onde ocorre a aplicação da regra operacional de defluências fixas, é apresentado a seguir.

```
...

if Disp <= 0 then L := 0
   else
     if Disp <= (CapUtil - VEspera) then L := DefluFixo
     else
       begin
       Vertimento := Disp - (CapUtil - VEspera);
       if Vertimento < DefluFixoHm3 then L := DefluFixo
          else L := PCR.Hm3_m3_Intervalo(Vertimento);
       end;
PCR.AtribuiDefluvioPlanejado(dt, L);

...
```

Essa etapa inicia-se pela verificação da disponibilidade hídrica total avaliada na etapa anterior. Sendo esta igual a zero, a defluência planejada, armazenada temporariamente na variável L, também será zero.

No caso de existir disponibilidade hídrica (*Disp* maior que zero) e quando o seu valor for inferior ao limite estabelecido pelo volume de espera (também definido pelo usuário), será liberado do reservatório a defluência fixa que foi estipulada no arquivo de entrada, para esse período do ano. Caso contrário, a rotina calcula um vertimento ou liberação de água do reservatório que, no mínimo, atenda a defluência planejada para o período.

Cabe salientar que se existir disponibilidade hídrica e seu valor for menor que a defluência fixa estabelecida pelo usuário, será liberada toda a água disponível, conforme estabelece a Regra Padrão que já está incorporada ao modelo.

Por fim, utiliza-se o método *AtribuiDefluvioPlanejado*, para armazenar o valor da defluência planejada para o atual intervalo de simulação.

A parte da regra que trata do racionamento no atendimento da demanda ou defluência solicitada pelo usuário, quando não houver água disponível no reservatório, é aplicada pelo próprio modelo Propagar, através de seu programa principal.

Durante o balanço hídrico de um PC com reservatório, o modelo verifica a disponibilidade de água no intervalo de tempo e, se necessário, aplica tal regra de racionamento. Sendo este um procedimento interno do modelo Propagar MOO, não será abordado neste trabalho.

Para que o leitor possa ter uma visão de conjunto dos “scripts” desenvolvidos no presente trabalho, todas as rotinas implementadas estão apresentadas no Anexo I.

3.2.2 Regra Padrão Modificada com Avaliação de Demandas

A chamada Regra Padrão Modificada é uma regra operacional baseada na regra padrão, abordada anteriormente. A sua seleção, para que fosse implementada como uma das regras operacionais genéricas no modelo Propagar MOO, se deve ao fato dela possuir uma metodologia de racionamento preventivo, cujo rigor é definido pelo próprio usuário, através de um parâmetro, conforme será visto em breve.

A regra também possui um outro parâmetro, definido pelo usuário, que estabelece um volume de espera no reservatório operado. No caso de ocorrer excesso hídrico em algum período, a regra estabelece um vertimento que, no mínimo, atende às demandas previstas.

Porém, pela forma como essa regra foi implementada, existe uma importante diferença em relação à regra anterior. A quantidade de água a ser liberada pelo reservatório não é definida diretamente pelo usuário, mas calculada pela própria rotina de planejamento que implementa a regra no modelo.

Para isso, a rotina realiza uma varredura em um trecho da rede hidrográfica, a jusante do reservatório, e avalia quais as demandas de água não são atendidas pelas afluências das próprias sub-bacias e trechos de rios à montante (demanda residual), totalizando-as e calculando qual deverá ser a defluência do reservatório naquele intervalo de tempo.

O que torna genérica a regra implementada nesse estudo é o fato dela poder ser aplicada, com mínimas alterações, em qualquer rede hidrográfica que contenha um único reservatório. Basta que o usuário informe em que ponto da rede está localizado o reservatório a ser operado, qual o ponto final do trecho da rede cujas demandas serão avaliadas e os parâmetros da regra operacional, tudo através de um arquivo tipo planilha de cálculo.

Os parâmetros da regra operacional, citados acima, são os que estabelecem o racionamento preventivo e o volume de espera para cada intervalo de tempo, ambos na forma de percentual do volume máximo do reservatório.

A Figura 3.3 representa a Regra Padrão Modificada, abordada por Vianna (1998), com seus parâmetros de racionamento preventivo e de formação de volume de espera.

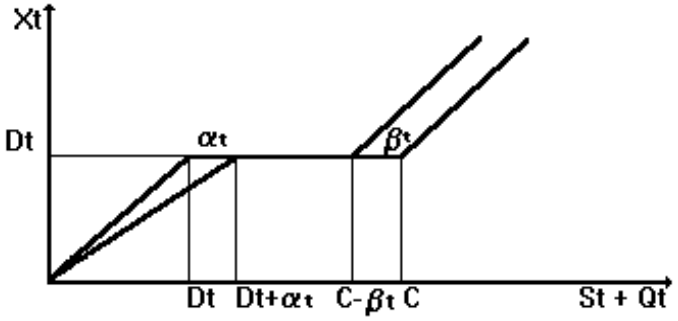


Figura 3.3: Regra padrão de decisão modificada.
(Fonte: Vianna Jr., 1998)

A regra padrão modificada utiliza um sistema de racionamento preventivo, caracterizado pela adoção do parâmetro $\mathbf{a_t}$ (Alfa), em unidades de volume, que é aplicado quando a disponibilidade hídrica no reservatório for menor que a soma entre a demanda residual ($\mathbf{D_t}$) e o parâmetro $\mathbf{a_t}$.

O primeiro trecho da curva na Figura 3.3, representa esse racionamento, onde nem toda água disponível será utilizada para atender a demanda residual $\mathbf{D_t}$. Essa situação se mantém até que a disponibilidade hídrica seja igual ou superior a $\mathbf{D_t + a_t}$, quando a demanda passa a ser integralmente atendida e ainda permanece no reservatório um volume $\mathbf{a_t}$. Vianna Jr. (1998) acrescenta que quanto maior for o valor de $\mathbf{a_t}$ mais rigoroso é o racionamento.

O parâmetro $\mathbf{b_t}$, também em unidades de volume, da regra modificada representa o procedimento de formação de um volume de espera no reservatório, para amortecimento de cheias. O valor da liberação de água será igual à demanda, até que a disponibilidade hídrica atinja o valor $\mathbf{(C - b_t)}$, quando é atingido o volume de espera.

A partir desse ponto será liberado um volume de água superior à demanda $\mathbf{D_t}$, fazendo com que o reservatório possa manter o volume de espera definido para esse período do ano. Vianna Jr. (1998) esclarece que quanto maior for o valor do parâmetro $\mathbf{b_t}$, maior será o volume de espera estabelecido pelo usuário. Observa-se que se os valores dos parâmetros $\mathbf{a_t}$ e $\mathbf{b_t}$ forem nulos, o esquema torna-se igual ao da regra padrão original.

Para melhor entendimento da estrutura conceitual da rotina construída para implementar a regra operacional descrita acima, é apresentado na Figura 3.4 um fluxograma das principais etapas da rotina de planejamento que utiliza a regra padrão modificada com avaliação de demandas.

Na primeira etapa da rotina ocorre a leitura do arquivo de dados criado pelo usuário do modelo. Este arquivo deve ser editado no padrão Excel 4.0, pelas razões já citadas, e ter o formato interno de organização de seus dados conforme o exemplo a seguir.

Tabela 3.2 – Exemplo de formato do arquivo de entrada de dados para a rotina de planejamento que aplica a Regra Padrão Modificada.

PC_8	PC_15										
11	12	13	14	15	16	16	15	14	13	12	11
5	6	7	8	9	10	10	9	8	7	6	5

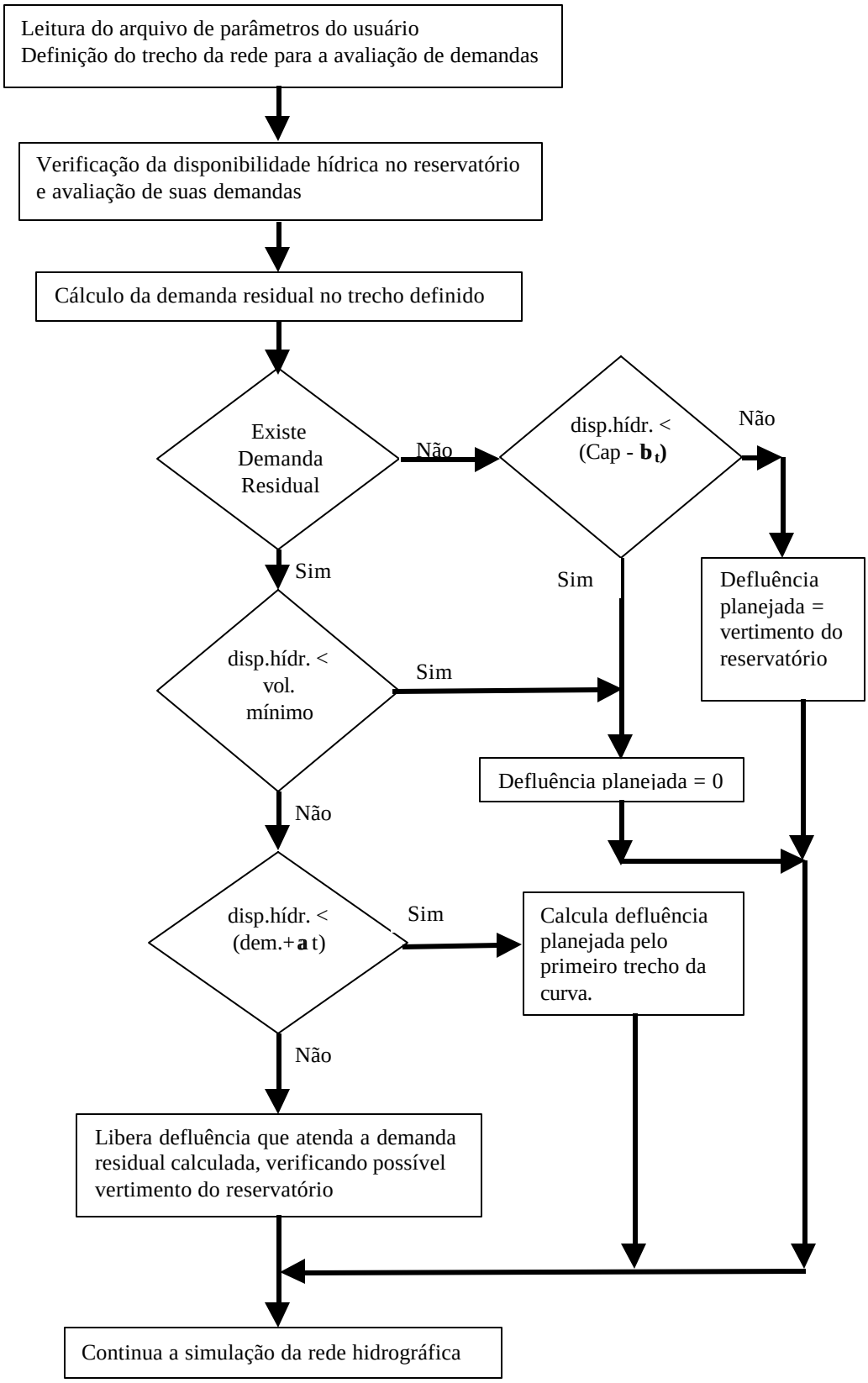


Figura 3.4: Fluxograma básico da Regra Padrão Modificada com Avaliação de Demandas.

A primeira célula, na primeira linha da planilha, deve conter a identificação do PC onde se localiza o reservatório a ser operado. A segunda célula dessa linha deve conter a identificação do PC, a jusante do reservatório, que finaliza o trecho da rede que deverá ser analisado para avaliar as demandas a serem atendidas pelo reservatório.

As primeiras 12 células da segunda linha devem conter os parâmetros mensais de racionamento preventivo ($\mathbf{a}_t$), na forma de percentual do volume máximo do reservatório.

As primeiras 12 células da terceira linha devem conter os parâmetros mensais que definem o volume de espera para cada mês do ano ($\mathbf{b}_t$), também na forma de percentual do volume máximo do reservatório.

O trecho da rotina em Pascal Script, que realiza essa etapa é demonstrado a seguir.

```
...

Plan := CreateObject(TPlanilha);
Plan.LoadFromFile('D:\Diretório\Arquivo.xls');
PCReserv := Plan.GetEntry(1,1);
PCDemandas := Plan.GetEntry(1,2);
vAlfa := Plan.RowToVec(2,1,12);
vBeta := Plan.RowToVec(3,1,12);
vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);

...
```

Inicialmente é criado um objeto tipo planilha, para o qual, logo a seguir, são copiados os dados fornecidos pelo usuário através de arquivo padrão Excel 4.0. As informações da planilha são copiadas para variáveis e objetos tipo vetor, de onde serão utilizadas pela rotina de planejamento.

Usando as informações de identificação dos PCs que delimitam o trecho definido pelo usuário, para avaliação das demandas a serem atendidas pelo reservatório, é criado um vetor – *vListaPCs*, na rotina acima – que contém uma lista dos PCs localizados entre *PCReserv* e *PCDemandas*, dentro da rede hidrográfica. No caso do exemplo da Tabela 3.2, PC_8 e PC_15.

Isso é feito através da função *PcsEntreDois*, localizada na última linha do trecho de rotina acima, que foi desenvolvida durante este trabalho, passando a ser uma importante contribuição para o modelo Propagar MOO. Essa função é abordada em detalhes no Anexo IV do presente trabalho.

A verificação da disponibilidade hídrica do reservatório é realizada de maneira similar à rotina que implementa a regra de defluências fixas, abordada anteriormente, sendo importante salientar que nessa segunda rotina de planejamento, também não são consideradas a precipitação e a evaporação sobre a superfície livre do reservatório.

Para iniciar o procedimento de avaliação das demandas, foi implementada uma etapa que totaliza as demandas no PC reservatório, apresentada abaixo.

```
...

DemReservatorio := 0;
For II :=1 to 3 do
   DemReservatorio := DemReservatorio + PCR.ObtemValorDemanda(dt, II, ‘T’);
DemReservatorio := PCR.m3_Hm3_Intervalo(DemReservatorio);

...
```

No trecho de rotina acima, observa-se o uso de um pequeno laço para somar as demandas primárias, secundárias e terciárias do PC reservatório, nesse intervalo de tempo, depositando o resultado na variável *DemReservatorio*. Nesse passo utilizou-se a função *ObtemValorDemanda*, com o auxílio da variável II. O resultado final é logo transformado de m^3/s para Hm^3/dt.

O trecho da rotina que calcula a demanda residual, ou seja, a demanda de água não atendida pelas afluências das próprias sub-bacias e de trechos de rios à montante, é apresentado a seguir.

```
...
N := vListaPCs.Count;
Aux := 0;
for I := 1 to (N-1) do
  begin
  PC    := TprPCP(vListaPCs.GetObject(I));
  Dem1 := PC.ObtemValorDemanda(dt, 1, 'T');
  Dem2 := PC.ObtemValorDemanda(dt, 2, 'T');
  Dem3 := PC.ObtemValorDemanda(dt, 3, 'T');
  SB   := PC.ObtemVazaoAfluenteSBs;
  PCAnt := TprPCP(vListaPCs.GetObject(I-1));
  DemAnt1 := PCAnt.ObtemValorDemanda(dt, 1, 'T');
  DemAnt2 := PCAnt.ObtemValorDemanda(dt, 2, 'T');
  DemAnt3 := PCAnt.ObtemValorDemanda(dt, 3, 'T');
  RetAnt1 := PCAnt.FatorDeRetorno(1);
  RetAnt2 := PCAnt.FatorDeRetorno(2);
  RetAnt3 := PCAnt.FatorDeRetorno(3);
  VazaoPCsMontante := 0;
  if PC.PCs_aMontante > 1 then
   begin
   for II := 0 to (PC.PCs_aMontante - 1) do
    begin
    PCMontante := PC.PC_aMontante(II);
```

```
    if PCMontante <> PCAnt then
     begin
     Projeto.RealizaBalancoHidricoAte(PCMontante);
     VazaoPCsMontante := VazaoPCsMontante + PCMontante.ObtemDefluencia(dt);
     end;
    end;
   end;
DemResidual := DemResidual + (Dem1 + Dem2 + Dem3) - SB - VazaoPCsMontante
             - (DemAnt1 * RetAnt1 + DemAnt2 * RetAnt2 + DemAnt3 * RetAnt3);
Aux := Max(Aux,Max(DemResidual,0));
end;
DemResidual := Aux;
DemResidualHm3 := PC.m3_Hm3_Intervalo(DemResidual);
...
```

As linhas de programação apresentadas acima formam um grande laço ou “loop”, criado para efetuar uma varredura no conjunto de PCs localizados entre o reservatório e o PC final para avaliação de demandas (ambos informados pelo usuário). A identificação dos PCs que formam esse conjunto é obtida a partir do vetor *vListaPCs* criado pela função *PcsEntreDois*, apresentada anteriormente.

Para avaliar a demanda residual do trecho, calcula-se a demanda não atendida pelas afluências de sub-bacias e PCs a montante, em cada PC, através da equação a seguir.

$$DemResidual = DEM - SB - VazaoPCsMontante - RET \quad \text{(Eq. 3.1)}$$

Onde:

DemResidual - Demanda residual do PC em análise;

DEM - Demandas primárias, secundárias e terciárias neste PC;

SB - Afluências de sub-bacias que contribuem para este PC;

VazaoPCsMontante - Afluências de PCs a montante do PC em análise, fora da lista de PCs do vetor *vListaPCs*;

RET - Retorno das demandas primárias, secundárias e terciárias do PC anterior, dentro da lista de PCs do vetor *vListaPCs.*

Ainda dentro desse trecho da rotina, realiza-se uma busca na rede hidrográfica para avaliar a existência de possíveis PCs que estejam a montante do PC em análise, mas fora da lista contida no vetor *vListaPCs*, e efetua-se um balanço hídrico parcial sobre cada um dos PCs encontrados nessas condições para determinar suas defluências.

O procedimento do Pascal Script utilizado para realizar o balanço hídrico parcial da rede chama-se *RealizaBalancoHidricoAte* e já foi abordado nesse capítulo, sendo detalhado no Anexo IV do presente trabalho.

No final do laço de comandos apresentado anteriormente, existe uma linha de programação que utiliza a variável auxiliar *Aux* para garantir que nela seja armazenada a maior deficiência hídrica encontrada ao longo da seqüência de PCs entre o *PCReserv* e o *PCDemandas*, através de comparação realizada pela função *Max*.

Após o cálculo das demandas residuais a serem atendidas pelo deflúvio do reservatório, a rotina de planejamento realiza a aplicação da regra operacional. O trecho da rotina que implementa esta etapa é demonstrado a seguir.

```
...

Projeto.DeltaT_ComoData(dt, Mes, Ano);
Alfa := PCR.VolumeMaximo * (vAlfa.Get(Mes))/100;
Beta := PCR.VolumeMaximo * (vBeta.Get(Mes))/100;
if DemResidual <= 0 then
  begin
  if (Disp – DemReservatorio) <= (CapUtil - Beta) then L := 0
  else L := PCR.Hm3_m3_Intervalo(Disp - DemReservatorio - CapUtil - Beta);
  end
else
  if (Disp – DemReservatorio) <= PCR.VolumeMinimo then L := 0
  else
    if (Disp – DemReservatorio) < (DemResidualHm3 + Alfa) then
      L := PCR.Hm3_m3_Intervalo((Disp – DemReservatorio) * DemResidualHm3 /
           (DemResidualHm3 + Alfa))
    else
      if (Disp – DemReservatorio) <= (CapUtil - Beta) then L := DemResidual
      else
        begin
        Vertimento := (Disp – DemReservatorio) - (CapUtil - Beta);
        if Vertimento < DemResidualHm3 then L := DemResidual
        else L := PCR.Hm3_m3_Intervalo(Vertimento);
        end;
PCR.AtribuiDefluvioPlanejado(dt, L);

...
```

Antes de fazer a aplicação da regra operacional, propriamente dita, realiza-se uma conversão dos valores dos parâmetros fornecidos pelo usuário, armazenados nos vetores vAlfa e vBeta na forma de percentual do volume máximo do reservatório. Isso é feito para converter os parâmetros Alfa e Beta em valores absolutos de volume (em Hm^3), adequando-os para utilização na aplicação da regra.

A etapa de aplicação da regra operacional, é constituída de um conjunto de comandos condicionais. O primeiro deles avalia a demanda residual, que, sendo nula, faz a rotina verificar a necessidade de vertimento do reservatório para cumprir a regra operacional, definindo a liberação de água igual ao possível vertimento ou igual a zero, conforme o caso.

Observa-se que, nessa etapa, a disponibilidade hídrica do reservatório é sempre usada em conjunto com a subtração das demandas do próprio reservatório. O objetivo é contemplar o atendimento das demandas de água desse PC, ao aplicar-se a regra operacional

Se existir demanda residual a ser atendida e, também, existir água disponível, a rotina irá executar outros comandos condicionais para avaliar em que ponto da curva, que representa a regra operacional, se encontra a disponibilidade hídrica do PC reservatório. O objetivo é definir qual a defluência a ser liberada do mesmo, nesse intervalo de tempo, para cumprir o que determina essa regra operacional.

3.2.3 Regra de Volumes-Metas com Avaliação de Demandas

Da forma como foi implementada nesse estudo, esta regra operacional busca controlar a defluência de um reservatório ao longo do tempo, de forma a atender as demandas de um determinado grupo de PCs a jusante, respeitando restrições de volumes mínimos e máximos do reservatório ao longo do ano.

O usuário define o conjunto de PCs a jusante do reservatório, cujas demandas serão avaliadas pela rotina, visando seu atendimento, bem como os valores dos volumes mínimos e máximos do reservatório para cada mês do ano. Essas informações são inseridas através de um arquivo tipo planilha de cálculo padrão Excel 4.0.

Os valores de volumes-metas mínimos e máximos do reservatório ao longo do ano, ao serem apresentados na forma de um gráfico, formam as chamadas curvas-guia de volumes, conforme exemplificado na Figura 3.5.

Limitando o volume do reservatório em um valor máximo, a regra operacional cria um volume de espera para controle de cheias. Dessa forma pode-se implementar volumes de espera adequados para cada período do ano.

Ao estabelecer um volume mínimo para o reservatório, a regra realiza um tipo de racionamento preventivo sobre o uso da água do reservatório, pois evita que um determinado volume de água seja utilizado no presente, preservando-o para utilização futura. Pode-se, assim, planejar as reservas mínimas a serem mantidas no reservatório em cada mês do ano.

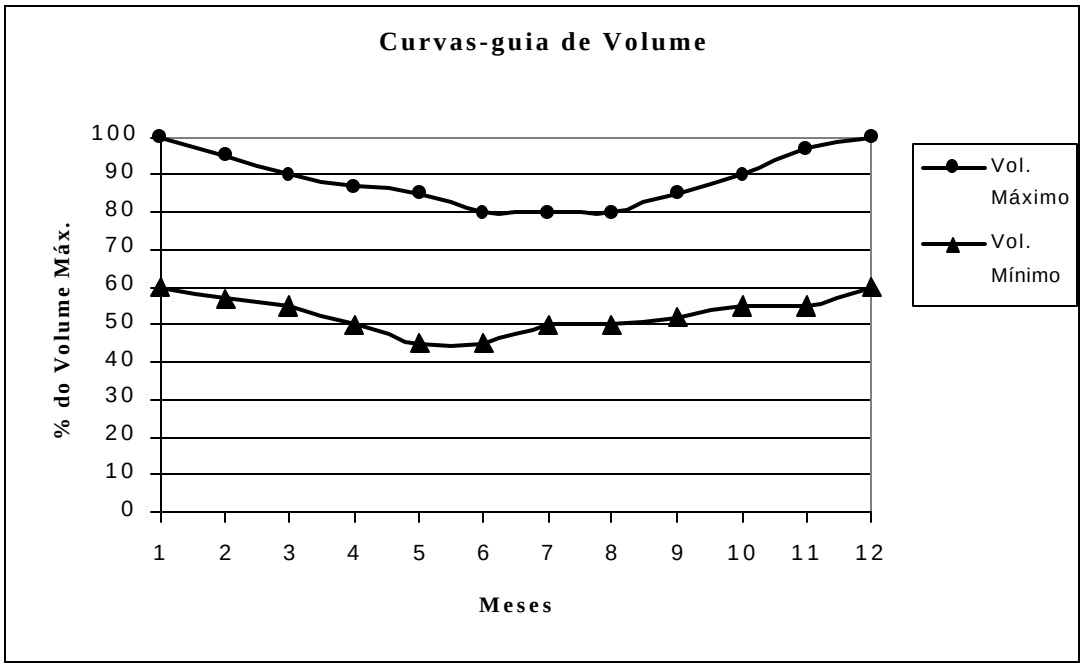


Figura 3.5 – Exemplo de Curvas-guia de Volume para operação de reservatórios.

Quando aplicada, essa regra operacional realizará um vertimento do excesso hídrico, quando o volume máximo definido para o período for ultrapassado, e provocará um racionamento nas demandas quando o volume mínimo estabelecido for atingido.

Para melhor entendimento do funcionamento da rotina criada para implementar essa regra operacional, apresenta-se na Figura 3.6, um fluxograma com suas principais etapas.

A etapa de entrada de dados do usuário segue o padrão utilizado nas rotinas anteriormente abordadas, sendo apresentada a seguir.

```
...
Plan := CreateObject(TPlanilha);
Plan.LoadFromFile('D:\Diretório\Arquivo.xls');
PCReserv := Plan.GetEntry(1,1);
PCDemandas := Plan.GetEntry(1,2);
vVolMetMax := Plan.RowToVec(3,1,12);
vVolMetMin := Plan.RowToVec(4,1,12);
vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);
...
```

As informações requeridas pela rotina são: identificação do PC onde se localiza o reservatório, identificação do último PC do trecho cujas demandas serão avaliadas, 12 valores mensais para o volume máximo e 12 valores mensais para o volume mínimo do reservatório.

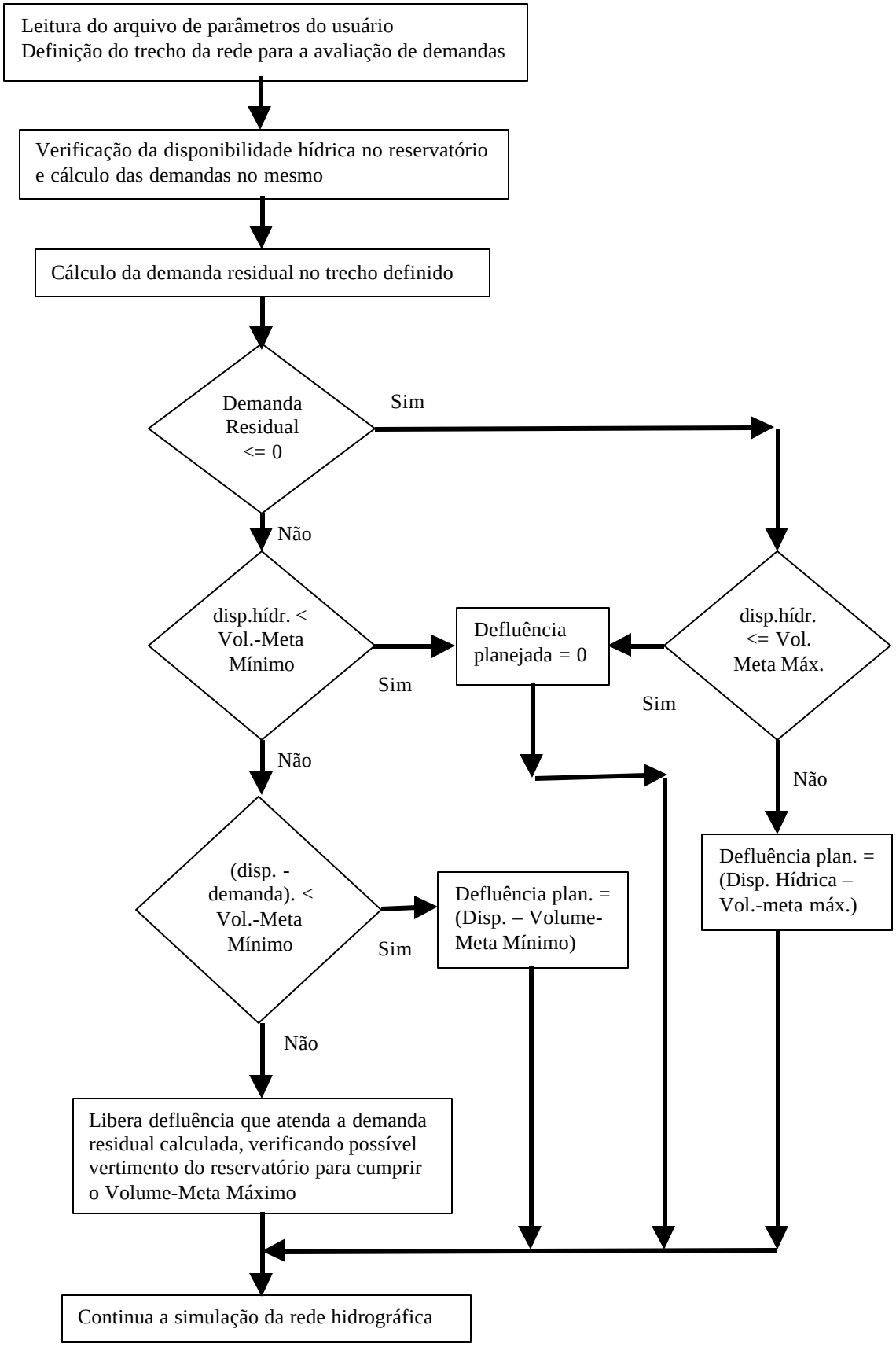


Figura 3.6: Fluxograma básico da regra de Volumes-Metas com Avaliação de Demandas.

Para que a rotina faça a leitura correta dos dados acima citados, é necessário criar o arquivo de entrada de dados com a estrutura interna apresentada na Tabela 3.3:

Tabela 3.3 – Exemplo de estrutura interna do arquivo de entrada de dados para a rotina de planejamento que aplica a regra de Volumes-metas.

PC_8	PC_10										
Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
100	100	90	80	80	70	70	75	80	90	95	100
10	10	10	15	15	15	15	15	15	10	10	10

Na primeira linha da planilha são colocadas as identificações dos PCs da rede hidrográfica, que representam, respectivamente, o reservatório que será operado e o ponto que finaliza o trecho para avaliação de demandas.

Na terceira linha da planilha são dispostos os valores que definem o volume-meta máximo do reservatório para cada mês do ano, na forma de percentual da capacidade máxima do reservatório.

Na quarta linha são dispostos os valores que definem o volume-meta mínimo para cada mês do ano, também na forma de percentual da capacidade máxima do reservatório.

A etapa da rotina que realiza a verificação da disponibilidade hídrica do reservatório foi implementada de forma idêntica à da rotina anterior, logo, entende-se não ser necessária aqui, uma abordagem detalhada.

Da mesma forma, os trechos da rotina que calculam a demanda total no PC reservatório e a demanda residual do conjunto de PCs definido pelo usuário (trecho usado para avaliar as demandas a serem atendidas pelo reservatório) têm as mesmas estruturas das respectivas etapas de cálculo adotadas na rotina de planejamento anterior, não sendo necessário reapresentá-las.

A etapa, cujos comandos e funções são apresentados a seguir é responsável pela aplicação da regra operacional, propriamente dita.

```
...

Projeto.DeltaT_ComoData(dt, Mes, Ano);
VolMetMax := PCR.VolumeMaximo*vVolMetMax.Get(Mes)/100 - PCR.VolumeMinimo;
VolMetMin := PCR.VolumeMaximo*vVolMetMin.Get(Mes)/100 - PCR.VolumeMinimo;
```

```
if VolMetMin < 0 then VolMetMin := 0;
if DemResidual <= 0 then
  begin
  if (Disp – DemReservatorio) <= VolMetMax then L := 0
    else
      begin
      Vertimento := ((Disp – DemReservatorio) - VolMetMax);
      L := PCR.Hm3_m3_Intervalo(Vertimento);
      end;
  end
else
  if (Disp – DemReservatorio) < VolMetMin then L := 0
  else
    if ((Disp – DemReservatorio) - DemResidualHm3) < VolMetMin then
      L := PCR.Hm3_m3_Intervalo(((Disp – DemReservatorio) - VolMetMin))
    else
      if (Disp – DemReservatorio) <= VolMetMax then L := DemResidual
      else
        begin
        Vertimento := (Disp – DemReservatorio) - VolMetMax;
        if Vertimento < DemResidualHm3 then L := DemResidual
        else L := PCR.Hm3_m3_Intervalo(Vertimento);
        end;
PCR.AtribuiDefluvioPlanejado(dt, L);

...
```

Nas primeiras linhas da rotina são feitas as transformações dos valores dos volumes-metas, de percentual da capacidade máxima do reservatório para volume absoluto do reservatório. Também é subtraído o volume morto do reservatório, ficando disponível nas variáveis de cálculo somente o volume útil do mesmo.

Após isso, verifica-se a existência de demandas residuais a serem atendidas pelo reservatório. Em caso negativo, a rotina apenas realiza o cumprimento da meta de volume máximo para o mês em simulação. Se existirem demandas residuais, a rotina verifica se o volume do reservatório possibilita o seu atendimento, sem descumprir a meta de volume mínimo. Observa-se que só será liberada a quantidade de água que não comprometa o atendimento da meta de volume mínimo do reservatório, estabelecida pelo usuário.

Se a disponibilidade hídrica superar a meta de volume máximo, a rotina planejará um vertimento, visando atender a essa meta. Nessa etapa, a disponibilidade hídrica é sempre acompanhada da subtração das demandas do próprio reservatório, para considerá-las no processo de planejamento.

Por fim, a defluência definida para o intervalo de tempo em análise é atribuída à propriedade de deflúvio planejado do PC onde se localiza o reservatório.

3.2.4 Regra de Zoneamento de Reservatório com Avaliação de Demandas

A regra operacional de zoneamento de reservatório, assim como a regra de planejamento apresentada anteriormente, propõe um tipo de racionamento das demandas associado ao volume de água do reservatório.

A regra de zoneamento de reservatório é indicada, na literatura, para sistemas com mais de um reservatório. Porém, para efeito de demonstração dessa ferramenta, aplicada ao modelo Propagar MOO, a regra foi implementada nesse estudo considerando a existência de somente um reservatório a ser operado.

Essa regra operacional consiste na divisão do volume útil do reservatório em um determinado número de zonas, limitadas por níveis estratégicos, às quais está vinculado o atendimento das demandas.

Supondo que um reservatório tenha o seu volume útil dividido em três zonas, conforme ilustra a Figura 3.7, o atendimento das demandas de jusante poderia ser realizado da seguinte maneira:

- Quando o volume do reservatório estiver dentro da zona 3, todas as demandas seriam atendidas;

- Quando o volume do reservatório estiver dentro da zona 2, seriam atendidas apenas as demandas consideradas como primárias ou secundárias, no trecho a jusante do reservatório; e

- Quando o volume do reservatório estiver dentro da zona 1, somente seriam atendidas as demandas primárias do trecho de jusante.

A regra determina que se o volume do reservatório for inferior ao volume morto, nenhuma demanda será atendida. Também permite que seja definido um volume de espera para o reservatório, possibilitando simular o controle de cheias.

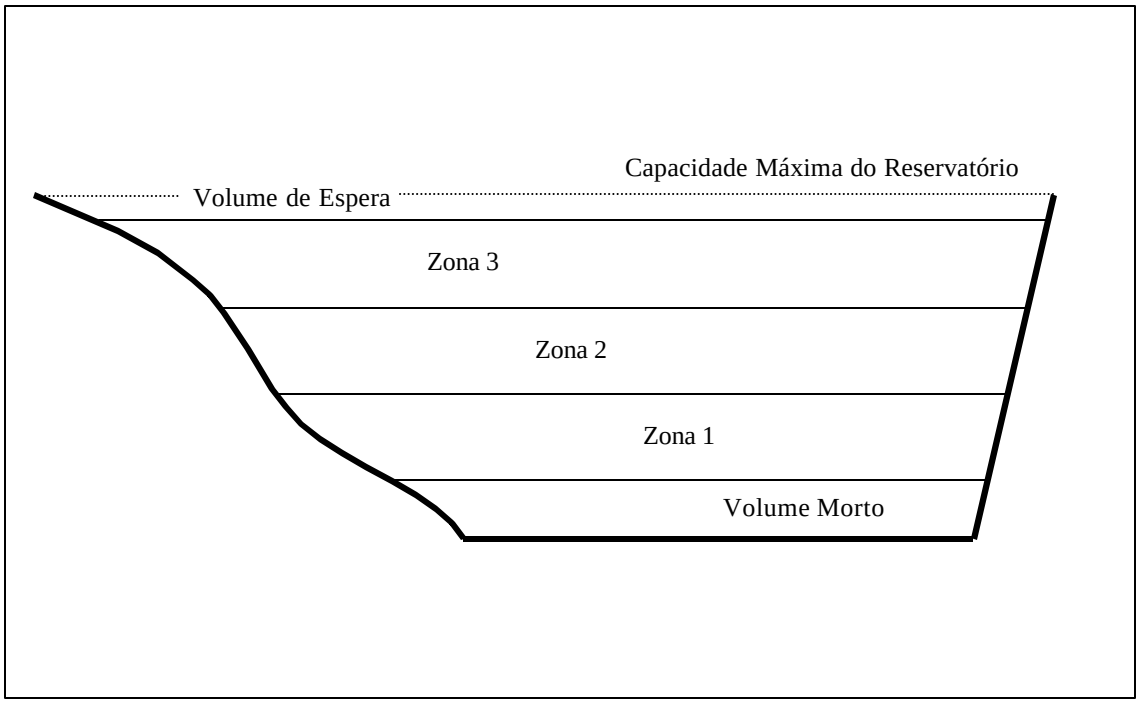


Figura 3.7: Exemplo de zoneamento de um reservatório.

No presente trabalho, a regra de zoneamento de reservatório foi implementada através de uma rotina em Pascal Script, de forma que o usuário possa definir a localização do reservatório dentro de uma rede hidrográfica, bem como o conjunto de PCs que definem o trecho da rede, a jusante do reservatório, cujas demandas procura-se atender. A própria rotina calcula as demandas a serem atendidas pelo reservatório.

Na rotina implementada neste estudo, adotou-se um número de três zonas para a divisão do volume útil do reservatório, conforme o que é ilustrado pela Figura 3.7. Os valores dos volumes do reservatório que definem os limites das zonas para atendimento das deferentes demandas, para cada mês do ano, são definidos pelo usuário.

Todas as informações estabelecidas pelo usuário são inseridas através de um arquivo tipo planilha de cálculo padrão Excel 4.0.

Visando o melhor entendimento do funcionamento da rotina que implementa a regra de zoneamento de reservatório, apresenta-se na Figura 3.8, um fluxograma das principais etapas da rotina.

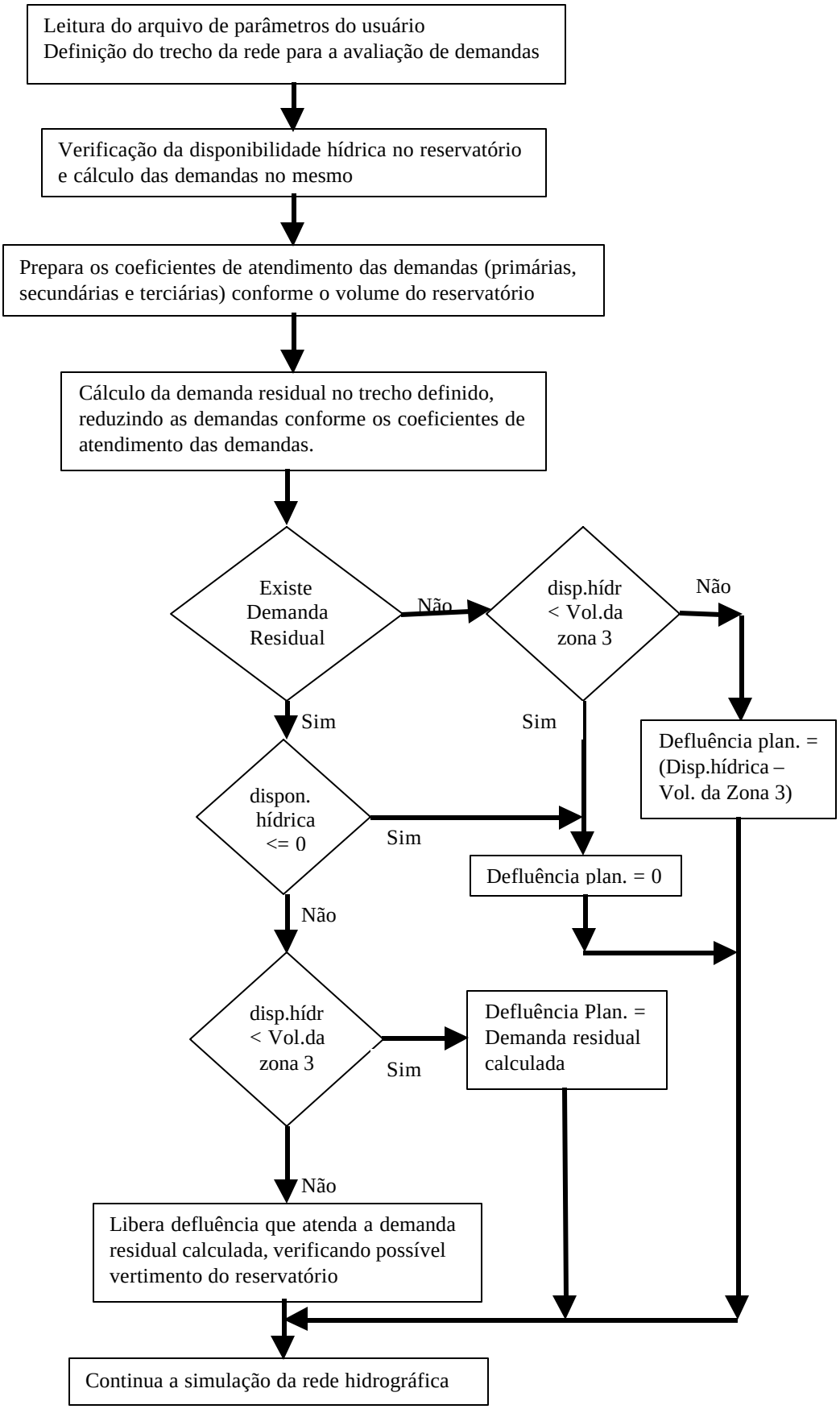


Figura 3.8: Fluxograma básico da regra de Zoneamento de Reservatório com Avaliação de Demandas.

Para realizar a etapa de entrada dos dados fornecidos pelo usuário, a rotina utiliza o seguinte conjunto de comandos em Pascal Script:

```
...

Plan := CreateObject(TPlanilha);
Plan.LoadFromFile('D:\Diretório\Arquivo.xls');
PCReserv := Plan.GetEntry(1,1);
PCDemandas := Plan.GetEntry(1,2);
vVolume1 := Plan.RowToVec(2,1,12);
vVolume2 := Plan.RowToVec(3,1,12);
vVolume3 := Plan.RowToVec(4,1,12);
vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);

...
```

As informações que o usuário deve fornecer para que a rotina opere corretamente são: a identificação do PC onde se localiza o reservatório, a identificação do PC que finaliza o trecho da rede cujas demandas devem ser atendidas pelo reservatório, 12 valores mensais para o parâmetro volume limite superior da zona 1, 12 valores mensais para o parâmetro volume limite superior da zona 2 e um conjunto de 12 valores mensais para o parâmetro volume limite da zona 3 do reservatório ao longo do ano. Todos esses valores devem ser introduzidos na forma de percentual da capacidade máxima do reservatório.

A tabela 3.4 apresenta um exemplo de como deve ser a organização interna desses dados dentro do arquivo tipo planilha, padrão Excel 4.0, elaborado pelo usuário.

Tabela 3.4 – Exemplo de estrutura interna do arquivo de entrada de dados para a rotina de planejamento que aplica a regra de Zoneamento de Reservatório.

PC_8	PC_15										
21	22	23	24	25	26	26	25	24	23	22	21
61	62	63	64	65	66	66	65	64	63	62	61
91	92	93	94	95	96	96	95	94	93	92	91

Na primeira linha devem estar as identificações dos PCs que limitam o trecho da rede para a análise de demandas residuais, sendo o primeiro deles o reservatório.

Nas linhas seguintes devem estar os valores mensais que representam os volumes que definem os limites superiores das zonas 1, 2 e 3, na forma de percentual do volume máximo do reservatório, sendo uma linha para os limites de cada zona.

A verificação da disponibilidade hídrica do reservatório e o cálculo de suas demandas foram implementados da mesma forma que nas rotinas anteriores, o que torna desnecessária uma abordagem específica.

As demandas, no modelo Propagar MOO, são classificadas em primárias, secundárias e terciárias, segundo a sua prioridade de atendimento. A definição da classe de prioridade de cada demanda inserida na rede hidrográfica é feita pelo usuário.

Para aplicar o racionamento das demandas conforme o volume do reservatório (regra de zoneamento), adotou-se um sistema de coeficientes de atendimento de demandas, onde cada coeficiente está relacionado com um tipo de demandas.

Assim foram criados três coeficientes de atendimento de demandas, *CoefPri*, *CoefSec* e *CoefTer*, cujos valores podem ser zero ou 1. Quando o valor de um coeficiente for igual a 1 (um), as demandas relacionadas a ele serão integralmente avaliadas para possível atendimento. Quando o seu valor for zero, as demandas planejadas relacionadas a ele serão anuladas, ou seja, não serão avaliadas pela etapa da rotina que calcula as demandas residuais.

A seguir apresenta-se o trecho da rotina que define os valores dos coeficientes de atendimento das demandas conforme o volume do reservatório.

```
...

Projeto.DeltaT_ComoData(dt, Mes, Ano);
Volume1 := PCR.VolumeMaximo*(vVolume1.Get(Mes))/100 – PCR.VolumeMinimo;
Volume2 := PCR.VolumeMaximo*(vVolume2.Get(Mes))/100 – PCR.VolumeMinimo;
Volume3 := PCR.VolumeMaximo*(vVolume3.Get(Mes))/100 – PCR.VolumeMinimo;
if (Disp – DemReservatorio) >= Volume2 then
  begin
  CoefPri := 1;
  CoefSec := 1;
  CoefTer := 1;
  end
 else
   if (Disp – DemReservatorio) >= Volume1 then
     begin
     CoefPri := 1;
     CoefSec := 1;
     CoefTer := 0;
     end
   else
     begin
     CoefPri := 1;
     CoefSec := 0;
     CoefTer := 0;
     end;

...
```

No início deste trecho realiza-se a conversão dos volumes limites de cada zona do reservatório, de valores percentuais da capacidade máxima do reservatório para valores absolutos de volume.

A seguir é avaliada a disponibilidade hídrica do reservatório nesse intervalo de tempo, para definir os valores dos coeficientes de atendimento das demandas para cada caso. Observa-se que a disponibilidade hídrica é avaliada considerando o atendimento das demandas do próprio reservatório.

A etapa de avaliação das demandas residuais do conjunto de PCs definido pelo usuário foi construída de forma similar àquela aplicada na rotina anterior, porém com algumas alterações. O trecho da rotina que implementa esta etapa é apresentado a seguir.

```
...
N := vListaPCs.Count;
Aux := 0;
for I := 1 to (N-1) do
  begin
  PC := TprPCP(vListaPCs.GetObject(I));
  Dem1 := CoefPri * (PC.ObtemValorDemanda(dt, 1, 'T'));
  PC.AtribuiValorDemanda(dt, 1, 'P', Dem1);
  Dem2 := CoefSec * (PC.ObtemValorDemanda(dt, 2, 'T'));
  PC.AtribuiValorDemanda(dt, 2, 'P', Dem2);
  Dem3 := CoefTer * (PC.ObtemValorDemanda(dt, 3, 'T'));
  PC.AtribuiValorDemanda(dt, 3, 'P', Dem3);
  SB   := PC.ObtemVazaoAfluenteSBs;
  PCAnt := TprPCP(vListaPCs.GetObject(I-1));
  DemAnt1 := CoefPri * (PCAnt.ObtemValorDemanda(dt, 1, 'T'));
  DemAnt2 := CoefSec * (PCAnt.ObtemValorDemanda(dt, 2, 'T'));
  DemAnt3 := CoefTer * (PCAnt.ObtemValorDemanda(dt, 3, 'T'));
  RetAnt1 := PCAnt.FatorDeRetorno(1);
  RetAnt2 := PCAnt.FatorDeRetorno(2);
  RetAnt3 := PCAnt.FatorDeRetorno(3);
  VazaoPCsMontante := 0;
  if PC.PCs_aMontante > 1 then
   begin
   for II := 0 to (PC.PCs_aMontante - 1) do
     begin
     PCMontante := PC.PC_aMontante(II);
     if PCMontante <> PCAnt then
       begin
       Projeto.RealizaBalancoHidricoAte(PCMontante);
       VazaoPCsMontante:=VazaoPCsMontante + PCMontante.ObtemDefluencia(dt);
       end;
     end;
   end;
  DemResidual := DemResidual + (Dem1 + Dem2 + Dem3) - SB – VazaoPCsMontante
              -(DemAnt1 * RetAnt1 + DemAnt2 * RetAnt2 + DemAnt3 * RetAnt3);
  Aux := Max(Aux,Max(DemResidual,0));
  end;
DemResidual := Aux;
DemResidualHm3 := PC.m3_Hm3_Intervalo(DemResidual);
...
```

Essa etapa de cálculo das demandas residuais do conjunto de PCs definido pelo usuário, difere da etapa similar, que compõe a rotina que implementa a regra de volumes-metas, por incluir os coeficientes que podem promover o racionamento das demandas de um PC e por utilizar o método *AtribuiValorDemanda*, que atribui a demanda planejada à respectiva propriedade do PC em análise, após a aplicação do racionamento.

Como, durante a simulação da propagação da água através da rede hidrográfica, o modelo procura atender unicamente as demandas planejadas, a aplicação desse método garante o cumprimento do racionamento estipulado pela regra operacional.

Dessa forma, a aplicação da regra operacional de zoneamento de reservatório tem seu início na etapa demonstrada acima, sendo concluída na etapa seguinte.

A última etapa da rotina, demonstrada a seguir, contempla a verificação da disponibilidade hídrica do reservatório, com suas afluências, para planejar a defluência do reservatório em cada intervalo de simulação.

```
...
if DemResidual <= 0 then
  begin
  if (Disp – DemReservatorio) <= Volume3 then L := 0
  else
    begin
    Vertimento := (Disp – DemReservatorio) – Volume3;
    L := PCR.Hm3_m3_Intervalo(Vertimento);
    end;
  end
else
  if (Disp – DemReservatorio) <= PCR.VolumeMinimo then L := 0
  else
    if (Disp – DemReservatorio) <= Volume3 then L := DemResidual
    else
      begin
      Vertimento := (Disp – DemReservatorio) – Volume3;
      if Vertimento < DemResidualHm3 then L := DemResidual
        else L := PCR.Hm3_m3_Intervalo(Vertimento);
      end;
PCR.AtribuiDefluvioPlanejado(dt, L);
...
```

Observa-se que haverá defluência sempre que existirem demandas residuais a serem atendidas e se a disponibilidade hídrica for superior ao volume mínimo do reservatório.

Esta etapa também verifica se o volume limite superior da última zona é atingido, situação na qual se faz necessário planejar um vertimento do reservatório que, no mínimo, atenda às demandas residuais.

3.3 FERRAMENTAS PARA A SIMULAÇÃO DE GERAÇÃO DE ENERGIA HIDRELÉTRICA EM PONTOS DA REDE HIDROGRÁFICA

3.3.1 Procedimento Padrão para o Cálculo da Energia Gerada.

Conforme descrito no capítulo 2, o modelo Propagar permitia a simulação da existência de usinas hidrelétricas em PCs da rede hidrográfica com controle de reservatório, bem como, o cálculo da quantidade de energia gerada durante um período de simulação.

Durante a realização deste trabalho, foram desenvolvidas ferramentas para o modelo Propagar MOO, no que se refere à simulação da geração de energia.

Primeiramente, foi implementada a possibilidade de qualquer PC possuir unidade de geração de energia, inclusive PCs sem o controle de reservatório. Isso torna possível simular a presença de usinas a fio d'água e contabilizar a energia gerada por elas.

Também foi implementada, na estrutura interna do modelo, uma nova forma para calcular a geração de energia nos PCs que simulem a presença de usinas hidrelétricas. No novo cálculo são consideradas várias características de um sistema de geração de energia que se deseje simular, não contempladas pelas versões anteriores do modelo.

Para tal, foram definidas e implementadas novas propriedades para os objetos tipo PC, com e sem reservatório, cujos dados são fornecidos pelo usuário do modelo através de janelas especialmente elaboradas para essa finalidade. São elas: o rendimento do sistema de adução, o rendimento das turbinas, o rendimento dos geradores, a máxima vazão turbinável, a queda hidráulica da usina (ou a cota de jusante da usina, em caso de PC reservatório) e a demanda energética.

As novas janelas de entrada de dados do usuário e as informações requeridas por elas são apresentadas nas Figuras 3.9 e 3.10. O acesso às mesmas é feito através de um botão, implementado no interior das janelas "Dados de um PC" e "Dados de um Reservatório", que é habilitado quando o usuário indicar a existência de geração de energia no PC.

Para o caso de PCs sem reservatório, o acionamento desse botão ativa uma nova janela denominada "Diálogo_PC_Energia" (Figura 3.9), criada para receber as informações mencionadas acima, referentes ao sistema de geração de energia.

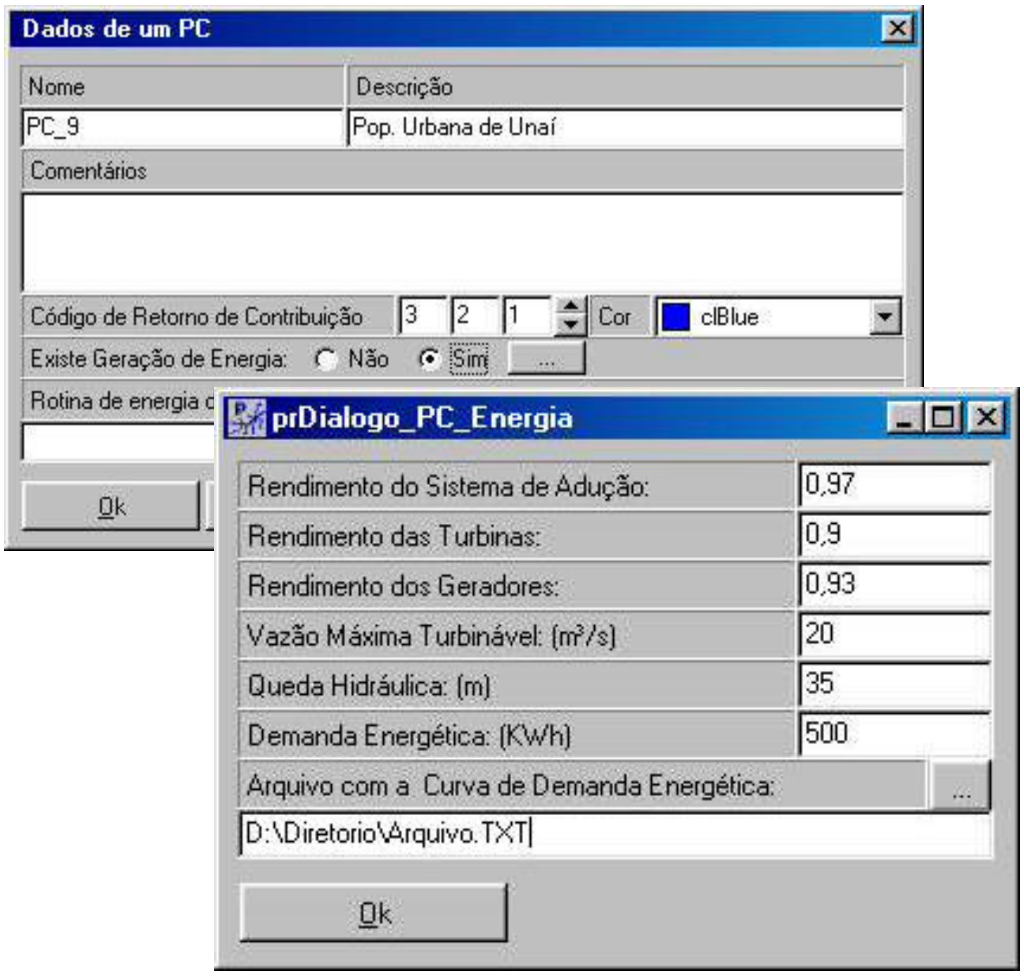


Figura 3.9: Janela de entrada de dados para a geração de energia em um PC.

O Rendimento do Sistema de Adução é utilizado para estimar as perdas de energia no trecho de adução, entre a tomada d'água da usina e a entrada da turbina.

Através da informação do Rendimento das Turbinas faz-se uma estimativa das perdas de energia que ocorrem internamente nas turbinas, enquanto que o Rendimento dos Geradores estima as perdas internas dos mesmos.

Os campos para inserção das informações de rendimento do sistema de adução, rendimento das turbinas e rendimento dos geradores são previamente preenchidos com os valores 0,97, 0,90 e 0,93, respectivamente, podendo ser alterados posteriormente. Isso é feito para auxiliar o usuário, em virtude desses serem plenamente compatíveis com os valores encontrados na literatura, para os respectivos parâmetros de rendimento.

A Vazão Máxima Turbinável é uma informação que será utilizada antes de ser calculada a energia gerada. Tem a função de limitar a vazão considerada no cálculo, representando, assim, as características do sistema real de geração.

A Queda Hidráulica da usina é uma informação específica de PCs que possuem usinas hidrelétricas sem reservatório de regularização, ou seja, usinas a fio d'água. Ela representa a altura de queda bruta, característica da usina que se deseja simular, dada em metros. Essa informação será utilizada diretamente no cálculo da geração de energia, que será abordado mais adiante.

As informações de Demanda Energética têm a função de disponibilizar ao usuário propriedades cujos dados representem a necessidade de energia a ser gerada nesta usina. Tais informações poderão ser utilizadas em rotinas Pascal Script, como por exemplo, rotinas de planejamento ou de cálculo de energia.

São disponibilizadas duas formas para o usuário inserir essas informações, através da referida janela. A primeira é utilizando diretamente o campo "Demanda Energética", onde o usuário pode digitar diretamente um valor fixo de energia, em kwh, que deva ser gerada nessa usina. A outra é indicando no campo inferior da janela, o nome de um arquivo de texto (.txt - padrão ASCII) que contenha 12 valores que representem a demanda de energia em cada mês, possibilitando ao usuário considerar uma demanda variável ao longo do ano. Ambas as formas têm suas informações armazenadas em propriedades do PC, independentes entre si, podendo ser utilizadas simultaneamente pelo usuário em rotinas Pascal Script, conforme sua necessidade.

No caso de geração de energia em um PC com reservatório, a janela de entrada de dados (Figura 3.10) apresenta uma pequena diferença em relação àquela usada para um PC sem reservatório de regularização: ao invés de ser solicitado a Queda Hidráulica, solicita-se a Cota de Jusante da Usina.

Essa alteração se deve ao fato de que, em usinas com reservatório, a altura de queda bruta varia ao longo do tempo, conforme a cota da água no reservatório. Para determinar a cota do reservatório em cada intervalo de tempo, utiliza-se uma função que, com o volume do reservatório e a sua curva cota-volume, calcula a cota correspondente.

Dessa forma, para estabelecer a altura de queda bruta da usina em cada intervalo de tempo, é necessário subtrair da cota calculada do reservatório a cota de jusante da usina. No presente trabalho, esta é considerada fixa, devendo ser fornecida pelo usuário.

As demais informações da janela são idênticas às solicitadas pela janela de dados de um PC sem reservatório, não sendo necessária nova abordagem.

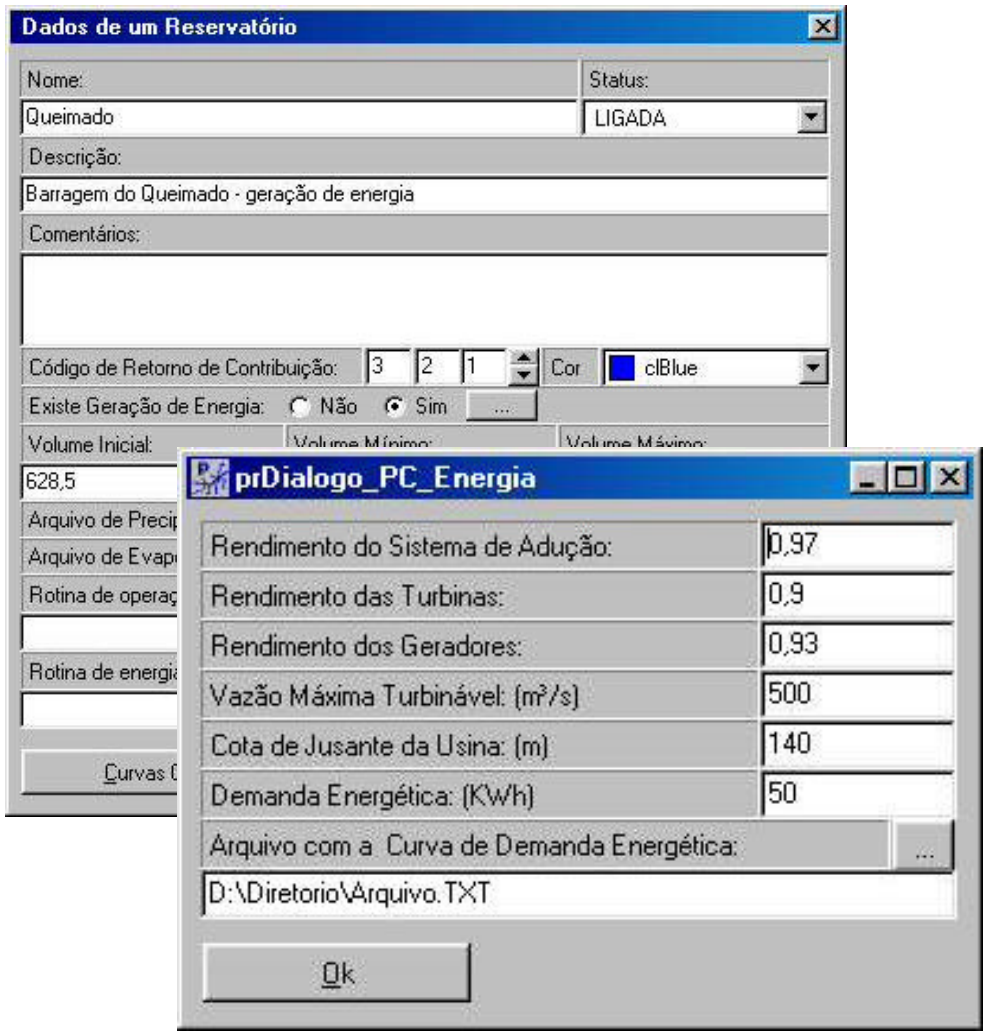


Figura 3.10: Janela de entrada de dados para a geração de energia em um reservatório.

As informações fornecidas pelo usuário são utilizadas na nova metodologia de cálculo da geração de energia implementada no modelo Propagar MOO, na forma apresentada a seguir.

No final do balanço hídrico realizado pelo modelo sobre cada PC, se este possuir usina hidrelétrica, é chamada uma função que calcula a quantidade de energia gerada pelo PC durante o intervalo de tempo. Os métodos que realizam o balanço hídrico de PCs com e sem reservatório são abordados no Anexo III desse trabalho.

A fórmula usada pela função para fazer o cálculo é a seguinte:

$$E = 9{,}81 \cdot h_A \cdot h_T \cdot h_G \cdot Q \cdot H \cdot Dt \qquad \text{(Eq. 3.2)}$$

Onde:

E – Energia gerada no PC durante o intervalo de tempo (kwh);
h_A – Rendimento da adução;
h_T – Rendimento médio das turbinas;
h_G – Rendimento médio dos geradores;
Q – Vazão média turbinada no período (m^3/s);
H – Altura de queda hidráulica média no período (m);
Dt – Intervalo de tempo ou período (horas);

Na aplicação da equação acima, os valores dos rendimentos são obtidos dos dados fornecidos pelo usuário, tanto para PC com reservatório, como para PC sem reservatório.

A vazão média turbinada no período é obtida durante o processo do balanço hídrico do PC, onde é calculada a vazão que efetivamente deflui desse PC, considerando uma possível vazão planejada. Porém, antes de adotar o valor extraído ao final do balanço hídrico, faz-se uma verificação e posterior ajuste, se necessário, para garantir o cumprimento da restrição de vazão máxima turbinável do sistema, informada pelo usuário.

A forma de obtenção do valor da altura de queda hidráulica média no período difere se o PC possui ou não reservatório, conforme já mencionado. No caso de PC sem reservatório, utiliza-se a "Queda Hidráulica da Usina", diretamente fornecida pelo usuário. Para o caso do PC possuir reservatório, utiliza-se o valor resultante da subtração entre a cota média do reservatório, calculada para cada intervalo de tempo, e a "Cota de Jusante da Usina", fornecida pelo usuário.

Por fim, o parâmetro intervalo de tempo refere-se ao número de horas contidas no intervalo de tempo adotado para a simulação.

O cálculo apresentado acima é realizado sempre que for executado o balanço hídrico de um PC que possua unidade de geração de energia. Por essa razão é denominado cálculo padrão.

3.3.2 Rotina de Cálculo de Energia.

O modelo Propagar MOO também possibilita que o usuário crie sua própria metodologia de cálculo através de uma rotina em Pascal Script. Dessa forma, o usuário pode alterar os parâmetros usados para o cálculo e até a forma de cálculo.

Na dinâmica do modelo Propagar, a metodologia padrão de cálculo sempre é executada, sendo, logo a seguir, chamada a rotina de cálculo de energia desenvolvida pelo usuário, se existir.

Para demonstrar o potencial do script de cálculo de energia criado pelo usuário, foi desenvolvida uma rotina que realiza o cálculo da energia gerada, em um PC com controle de reservatório, considerando o rendimento da turbina variável em função da vazão turbinada.

Essa rotina tem a função de quantificar, para cada intervalo de simulação, a energia gerada em um PC com reservatório, considerando o parâmetro Rendimento da Turbina variável em função da vazão turbinada, segundo uma curva fornecida pelo usuário.

Para auxiliar o entendimento da rotina elaborada, é apresentado na Figura 3.11, um fluxograma das principais etapas dessa rotina de cálculo de energia.

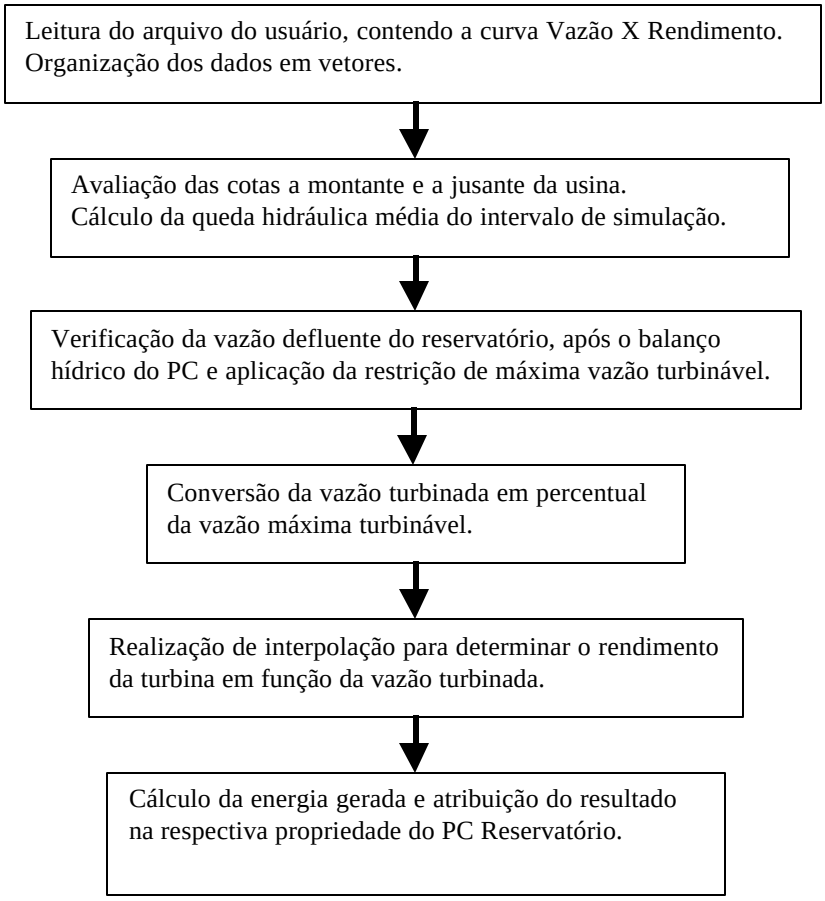


Figura 3.11: Fluxograma básico da rotina de Cálculo de Energia.

Na primeira etapa da rotina, ocorre a leitura da curva que relaciona o rendimento da turbina com a vazão turbinada. Esta etapa foi implementada de forma que a curva seja fornecida pelo usuário através de um arquivo tipo planilha de cálculo padrão Excel 4.0, cuja organização interna de dados esteja de acordo com o exemplo demonstrado na Tabela 3.5. Esse arquivo é lido diretamente através de comandos internos da rotina, conforme será visto posteriormente.

Tabela 3.5 – Exemplo de estrutura interna do arquivo de entrada de dados para a rotina que realiza o Cálculo de Energia.

PC_8	8
Q/Qmax (%)	Rendimento
10	65
20	75
40	80
60	85
70	90
80	91
90	90
100	88

Nesse exemplo, a primeira célula da primeira linha contém a identificação do PC da rede hidrográfica onde se localiza o reservatório com o sistema de geração de energia. Na segunda coluna, nessa mesma linha, encontra-se um valor que indica o número de pontos da curva que relaciona o rendimento da turbina com a vazão turbinada.

A curva “Rendimento X Vazão” é fornecida através de pontos, cujos valores são dispostos no arquivo em duas colunas, sendo a primeira referente à vazão turbinada e a segunda referente ao rendimento da turbina.

Cabe salientar que os valores referentes à vazão turbinada estão na forma de percentual da vazão máxima turbinável.

O trecho da rotina em Pascal Script que implementa a etapa referente à entrada de dados fornecidos pelo usuário é apresentada a seguir, onde observa-se que os dados referentes à curva são armazenados em vetores para futura utilização no cálculo.

```
...

Planilha := CreateObject(TPlanilha);
NomeArquivo := 'D:\Diretório\Arquivo.xls';
Planilha.LoadFromFile(NomeArquivo);
NomePCR := Planilha.GetEntry(1, 1);
NLV := Trunc(Planilha.GetFloat(1, 2));
NLP := NLV + 2;
NomeSerie := Planilha.GetEntry(2, 2);
vCurvaVazao := CreateObject(TwsDFVec, NLV);
vCurvaRendi := CreateObject(TwsDFVec, NLV);
vCurvaVazao := Planilha.ColToVec(1, 3, NLP);
vCurvaRendi := Planilha.ColToVec(2, 3, NLP);

...
```

A maioria dos comandos e funções utilizados nessa etapa já foram abordados em comentários sobre outros trechos de rotinas, apresentados anteriormente, porém, existe uma função usada na etapa acima que será descrita em detalhes. É a função *Trunc*, que retorna um número inteiro, obtido pelo "truncamento" de um número real.

Quando a informação referente ao número de linhas de dados da curva fornecida pelo usuário, localizada na primeira linha e segunda coluna da planilha, é obtida com o uso da função *GetFloat*, seu formato é convertido para o de um número real. Ocorre que essa informação será utilizada pela rotina como um número inteiro, justificando, assim, a necessidade da transformação realizada pela função *Trunc*.

A função *ColToVec* copia os dados de uma coluna da planilha para o interior de um vetor previamente criado.

Após a leitura dos dados em arquivo, realiza-se a etapa de avaliação das cotas da usina e do cálculo da queda hidráulica, sendo seus comandos apresentados logo a seguir.

```
...

CotaDoDt := PCR.CalculaCotaHidraulica((VolumeAtual+VolumeAnterior)/2);
CotaJusante := PCR.CotaJusante;
Queda := CotaDoDt - CotaJusante;

...
```

A cota do reservatório a montante da usina em cada intervalo de tempo, é obtida pela utilização da função *CalculaCotaHidraulica*, que aplica o volume médio do reservatório durante o intervalo sobre a curva cota-volume desse reservatório (inserida durante a construção da rede hidrográfica).

A queda hidráulica a ser considerada durante o intervalo é obtida pela subtração entre a cota de montante e a cota de jusante da usina, fornecida pelo usuário na janela de dados do sistema de geração de energia de um PC reservatório.

Na etapa seguinte, verifica-se a vazão defluente do reservatório, determinada no processo de balanço hídrico do PC, sendo seu valor convertido para m^3/s. Nessa etapa, a rotina também realiza a verificação e, se necessário, a correção para aplicar a restrição da máxima vazão turbinável. Os comandos dessa etapa são apresentados a seguir.

```
...

Vazao := PCR.ObtemDefluvioOperado(dt);
Vazao := PCR.Hm3_m3_Intervalo(Vazao);
VazaoMaxTurb := PCR.VazaoMaxTurbina;
if Vazao > VazaoMaxTurb then Vazao := VazaoMaxTurb;

...
```

Como os dados de vazão da curva “Rendimento x Vazão Utilizada”, estão na forma de percentual da máxima vazão turbinável, é necessário converter o valor da vazão operada pelo reservatório para o mesmo formato, tornando possível sua utilização no processo de interpolação. Isso é realizado pela instrução demonstrada a seguir.

```
...
B := (Vazao / VazaoMaxTurb) * 100;
...
```

Para obter o valor do rendimento da turbina referente à vazão turbinada, de acordo com a curva Vazão X Rendimento, utiliza-se um processo de interpolação linear. A etapa da rotina que implementa este processo é apresentada a seguir.

```
...
vAuxIndice := vCurvaVazao.FindGTE(B);
Indice := vAuxIndice.GetAsInteger(1);
FreeObject(vAuxIndice);
if Indice = 1 then
   begin
     A := 0;
     D := 0;
   end
  else
    begin
      A := vCurvaVazao.Get(Indice-1);
      D := vCurvaRendi.Get(Indice-1);
    end;
C := vCurvaVazao.Get(Indice);
E := vCurvaRendi.Get(Indice);
RendTurbina := ((E - D) * (Vazao - A) / (C - A) + D) / 100;
...
```

Na última etapa da rotina, realiza-se o cálculo da energia gerada no PC, nesse intervalo de simulação, utilizando os dados de rendimento de adução e rendimento de gerador, já inseridos pelo usuário. A fórmula de cálculo é a mesma utilizada internamente no modelo (cálculo padrão), não sendo necessária uma abordagem mais detalhada. Os comandos da rotina, que implementam essa etapa são apresentados abaixo

```
...
RendGerador := PCR.RendimentoGeradores;
RendAducao  := PCR.RendimentoAducao;
Potencia := 9.81 * RendAducao * RendTurbina * RendGerador * Vazao * Queda;
Energia := Potencia * (Projeto.DiasNoIntervalo * 24);
PCR.AtribuiEnergia(dt, Energia);
...
```

Mas, para que ocorra geração de energia em uma usina localizada em um determinado PC, é necessário que exista, de fato, vazão defluindo desse ponto da rede.

No caso desse ponto ser um reservatório, ou seja, uma usina hidrelétrica com reservatório de regularização, é preciso utilizar na simulação uma rotina que garanta a liberação de água para geração de energia.

A liberação de água pode ser obtida com a utilização de rotinas de planejamento que, por exemplo, estipulem valores fixos de defluência para o reservatório. Neste trabalho, já foi apresentada uma rotina de planejamento que permite ao usuário do modelo definir defluências fixas para um determinado reservatório, podendo ser utilizada como modelo.

Porém, é possível implementar algo mais elaborado, como por exemplo, uma rotina de planejamento que verifique qual a demanda de energia definida pelo usuário para essa usina e que determine qual a vazão a ser liberada para atender a essa demanda.

Para implementar esse exemplo, foi elaborada uma rotina de planejamento relativamente simples, utilizando o editor Pascal Script, cujos principais comandos e funções são apresentados abaixo.

```
...
Queda := (PCR.CalculaCotaHidraulica(VolInicInterv) - PCR.CotaJusante);
RendAducao  := PCR.RendimentoAducao;
RendTurbina := PCR.RendimentoTurbina;
RendGerador := PCR.RendimentoGeradores;
Projeto.DeltaT_ComoData(dt, Mes, Ano);
DemandaEnergia := PCR.CurvaDemandaEnergetica(Mes);
Potencia := (DemandaEnergia / (Projeto.DiasNoIntervalo * 24));
Vazao := Potencia /(9,81 * RendAducao * RendTurbina * RendGerador * Queda);
PCR.AtribuiDefluvioPlanejado(dt, Vazao);
...
```

Observa-se que a rotina utiliza as mesmas equações do cálculo padrão, usado internamente no modelo Propagar MOO, para determinar a vazão a ser liberada do reservatório em cada intervalo de simulação.

Através dessas ferramentas, demonstra-se que o usuário tem a possibilidade de criar e aplicar diferentes formas para estimar a quantidade de energia média gerada em um PC com usina, durante uma simulação com o modelo Propagar MOO.

3.4 FERRAMENTAS PARA ANÁLISE DOS RESULTADOS DE SIMULAÇÕES.

O modelo Propagar MOO possui algumas ferramentas para análise dos resultados de uma simulação, disponíveis, automaticamente, através dos seus menus. Dentre elas pode-se citar o relatório de falhas no atendimento das demandas, planilhas e gráficos que apresentam as propriedades de cada PC ao final da simulação.

A utilização de tais ferramentas permite, por exemplo, analisar as falhas no atendimento de demandas, além de outras verificações sobre as propriedades de cada PC da rede hidrográfica, em cada intervalo de tempo.

Entre as propriedades dos PC, acessíveis no final da simulação, pode-se citar: as afluências e defluências; as vazões de montante; as demandas atendidas, planejadas e totais; a energia gerada em um PC; ou, ainda, os volumes dos reservatórios ao longo da simulação.

Porém, o modelo Propagar MOO também permite que o usuário crie funções para análise dos resultados da simulação, através de rotinas programadas em Pascal Script.

Para isso, o modelo Propagar dispõe de uma chamada de rotina, denominada Rotina de Uso Geral, executada pelo modelo ao final de uma simulação (após o último intervalo de tempo de simulação).

A Rotina de Uso Geral permite que sejam implementadas ferramentas para auxiliar a análise de resultados, como por exemplo a construção de gráficos e tabelas. Seu uso também permite a inclusão de funções de rastreamento (debug) da execução, permitindo a realização de testes durante o processo de simulação.

Para mostrar a aplicabilidade da Rotina de Uso Geral e, ao mesmo tempo, demonstrar a construção de ferramentas para a análise dos resultados de uma simulação, foram desenvolvidas rotinas, usando a linguagem de programação Pascal Script, para serem utilizadas em conjunto com rotinas de planejamento do uso da água.

São elas: gráfico de volumes-metas com volume médio, curva de permanência da energia gerada e curva de energia média gerada.

3.4.1 Gráfico de Volumes-Metas com Volume Médio

A rotina de uso geral denominada “Gráfico de Volumes-Metas com Volume Médio” foi desenvolvida para ser utilizada em conjunto com a rotina de planejamento de Volumes-Metas, visando criar um gráfico que demonstrasse os volumes médios mensais de um reservatório, juntamente com os volumes-metas planejados pelo usuário.

Dessa forma, além do desenvolvimento da própria ferramenta, procura-se, também, demonstrar a metodologia para construção de gráficos utilizando o Pascal Script.

A primeira etapa dessa rotina, apresentada abaixo, realiza a leitura dos dados de volumes-metas estabelecidos pelo usuário, de forma similar à apresentada na rotina de planejamento e converte os dados da forma de percentual da capacidade máxima do reservatório para valores absolutos de volume.

```
...

Plan := CreateObject(TPlanilha);
Plan.LoadFromFile('D:\Diretório\Arquivo.xls');
PCReserv := Plan.GetEntry(1,1);
vVolMetMax := Plan.RowToVec(3,1,12);
vVolMetMin := Plan.RowToVec(4,1,12);
FreeObject(Plan);
PCR := Projeto.PCPeloNome(PCReserv);
for i := 1 to 12 do
  begin
  vVolMetMax.Set(i, vVolMetMax.Get(i) * PCR.VolumeMaximo / 100);
  vVolMetMin.Set(i, vVolMetMin.Get(i) * PCR.VolumeMaximo / 100);
  end;

...
```

Na etapa seguinte da rotina, são realizados os procedimentos de cálculo para determinar os volumes médios mensais, com a utilização de vetores auxiliares. Para otimizar a montagem da etapa, utilizou-se uma combinação de comandos de laço.

```
...

for i := 1 to 12 do
  begin
  vTempo.Set(i, i);
  for ii := 0 to (Projeto.NumAnosDeExecucao - 1) do
    begin
    NovoValor := vVolumes.Get(12*ii +i) / Projeto.NumAnosDeExecucao;
    NovoValor := vVolumesMedios.Get(i) + NovoValor;
    vVolumesMedios.Set(i, NovoValor);
    end;
  end;

...
```

A etapa final da rotina é responsável pela construção e apresentação do gráfico, sendo descrita e apresentada a seguir, iniciando pela definição das cores de cada elemento que será graficado, passando pela definição do tipo de gráfico e pelo título a ser apresentado. A seguir são definidas as três séries a serem apresentadas na forma de gráfico de barras, com seus respectivos rótulos (legendas). Por fim, são adicionados os valores das séries, a partir dos vetores previamente calculados e preenchidos nas etapas anteriores.

```
...

CorMetaMin := getColor('green');
CorVolume  := getColor('blue');
CorMetaMax := getColor('red');
Grafico := CreateObject(TgrGrafico);
Grafico.Chart.SetView3D(True);
Grafico.Chart.Title.Add('Volumes no Reservatório');
SerieVolMetMin := Grafico.Series.AddBarSerie('Mínimo', CorMetaMin, 0,1);
SerieVolumesMedios := Grafico.Series.AddBarSerie('Médios', CorVolume, 0,1);
SerieVolMetMax := Grafico.Series.AddBarSerie('Máximo', CorMetaMax, 0,1);
for i := 1 to 12 do
begin
SerieVolMetMin.AddXY(vTempo.Get(i), vVolMetMin.Get(i), '', CorMetaMin);
SerieVolumesMedios.AddXY(vTempo.Get(i),vVolumesMedios.Get(i),'',CorVolume);
SerieVolMetMax.AddXY(vTempo.Get(i), vVolMetMax.Get(i), '', CorMetaMax);
end;
Grafico.Show;

...
```

3.4.2 Curva de Permanência da Energia Gerada

Com o objetivo de implementar uma ferramenta para auxiliar a análise de resultados de uma simulação realizada para avaliar a geração de energia de uma usina hidrelétrica, criou-se uma rotina que determina a curva de permanência da energia gerada em um PC.

Essa rotina é dividida em quatro etapas: criação e preenchimento de vetores, cálculo da curva de permanência, construção e apresentação dos dados em planilha e construção e apresentação dos dados em um gráfico.

A primeira etapa dessa rotina, apresentada a seguir, realiza a criação e o preenchimento dos vetores utilizados no cálculo e na apresentação dos resultados.

```
...
vTempo := CreateObject(TwsSIVec, IntTotal);
vEnergiaPCR     := CreateObject(TwsDFVec, IntTotal);
vEnergiaPCRDecr := CreateObject(TwsDFVec, IntTotal);
vFreqPCR        := CreateObject(TwsDFVec, IntTotal);
for i := 1 to IntTotal do
  begin
  vTempo.Set(i, i);
  vEnergiaPCR.Set(i, PCR.ObtemEnergia(i));
  end;
...
```

Nessa etapa, o vetor *vEnergiaPCR* é preenchido com os dados de energia gerada no PC reservatório ao longo da simulação. Também é preparado o vetor *vTempo*.

Na seqüência, realiza-se a etapa de cálculo da curva de permanência, cujo trecho da rotina responsável por sua execução é apresentado a seguir.

```
...
for i := 1 to IntTotal do
  vEnergiaPCRDecr.Set(i, vEnergiaPCR.Get(i));
vEnergiaPCRDecr.Sort(false, false);
for i := 1 to IntTotal do
  vFreqPCR.Set(i, (100*i/IntTotal));
...
```

Essa etapa inicia pela transferência dos dados de energia gerada, localizados no vetor *vEnergiaPCR*, para o vetor auxiliar *vEnergiaPCRDecr*, através de um laço usando o comando *for*. Logo após, esse vetor auxiliar é organizando de maneira que seus dados sejam colocados de forma decrescente, através do método *Sort(false, false)*, dando origem a um dos vetores que será usado na construção da curva de permanência de energia gerada.

Na elaboração do outro vetor a ser utilizado na construção da referida curva (*vFreqPCR*), realiza-se um laço com o comando *for*, usando a própria variável de indexação dos elementos do vetor para gerar as informações de preenchimento do mesmo. Assim, ao ser utilizado um mesmo indexador para acessar os elementos de ambos os vetores, o conteúdo do vetor *vFreqPCR* representará as freqüências associadas aos valores de energia gerada.

Para apresentar os resultados do cálculo, adotou-se duas maneiras. A primeira é a construção de uma planilha, apresentada abaixo, e a segunda é a construção de um gráfico.

Para elaborar uma saída de dados tipo planilha, criou-se um objeto tipo planilha e foram usados basicamente os comandos *Write* e *WriteVecInCol*, para escrever os títulos dos dados em células e transferir os dados dos vetores para colunas da planilha, respectivamente.

```
...

Planilha := CreateObject(TPlanilha);
Planilha.Write(1, 1,'Curva de Permanência da Energia Gerada');
Planilha.Write(2, 1, 'Projeto ' + Projeto.Nome);
Planilha.Write(3, 1, 'Índice');
Planilha.Write(3, 2, 'Energia');
Planilha.Write(3, 3, 'Probabilidade');
Planilha.WriteVecInCol(vTempo, 1, 4);
Planilha.WriteVecInCol(vEnergiaPCRDecr, 2, 4);
Planilha.WriteVecInCol(vFreqPCR, 3, 4);
Planilha.Show;

...
```

Na etapa final da rotina, realiza-se a apresentação dos dados calculados na forma de um gráfico. É criado um objeto tipo gráfico, definido como gráfico de duas dimensões, sendo definido um título. A esse gráfico adiciona-se uma série tipo linha (gráfico de linhas), cujos dados são provenientes dos vetores calculados. No final do trecho está o método *Show*, que mostra o gráfico no monitor. Os comandos são apresentados a seguir.

```
...

Cor := getColor('Black');
Grafico := CreateObject(TgrGrafico);
Grafico.Chart.SetView3D(False);
Grafico.Chart.Title.Add('Curva de Permanência de Energia Gerada');
Serie1 := Grafico.Series.AddLineSerie('PCR', Cor);
for i := 1 to IntTotal do
   Serie1.AddXY(vFreqPCR.Get(i), vEnergiaPCRDecr.Get(i), '', Cor);
Grafico.Show;

...
```

3.4.3 Curva da Energia Média Gerada

O último script criado para demonstrar a aplicabilidade da rotina de uso geral, foi o que calcula e apresenta a relação das médias mensais de energia gerada, comparando-as com as demandas mensais de energia informadas pelo usuário.

Sua elaboração surgiu do interesse em criar uma outra ferramenta para auxiliar na análise de resultados de uma simulação com geração de energia e reforçar a demonstração da criação de gráficos e planilhas em Pascal Script.

Na sua primeira etapa, apresentada a seguir, a rotina cria e preenche os vetores que serão utilizados pela etapa de cálculo, com os dados de energia gerada em cada intervalo de tempo e a demanda de energia estabelecida pelo usuário para cada mês do ano.

```
...

vTempo := CreateObject(TwsSIVec, 12);
vDemEnergiaPCR := CreateObject(TwsDFVec, 12);
vEnergGeradaPCR  := CreateObject(TwsDFVec, IntTotal);
vEnergMediaPCR   := CreateObject(TwsDFVec, 12);
for i := 1 to 12 do
  begin
  vTempo.Set(i, i);
  vDemEnergiaPCR.Set(i, PCR.CurvaDemandaEnergetica(i));
  end;
for i := 1 to IntTotal do
  vEnergGeradaPCR.Set(i, PCR.ObtemEnergia(i));

...
```

Na etapa seguinte, aqui apresentada, calcula-se a energia média gerada em cada mês do ano. Para tal utiliza-se um conjunto de comandos tipo laço, similar ao utilizado em outra rotina anteriormente apresentada.

```
...

for i := 1 to 12 do
  for ii := 0 to (Projeto.NumAnosDeExecucao - 1) do
    begin
    NovoValor := vEnergGeradaPCR.Get(12*ii+i)/ Projeto.NumAnosDeExecucao;
    NovoValor := vEnergMediaPCR.Get(i) + NovoValor;
    vEnergMediaPCR.Set(i, NovoValor);
    end;

...
```

A etapa final dessa rotina é responsável pela apresentação dos dados calculados na forma de um gráfico de barras. Nesse gráfico são comparados os valores da energia média gerada em cada mês com a demanda estabelecida pelo usuário do modelo. Seus comandos são apresentados abaixo.

```
...

CorDemanda := getColor('red');
CorEnergia := getColor('blue');
Grafico := CreateObject(TgrGrafico);
Grafico.Chart.SetView3D(True);
Grafico.Chart.Title.Add('Curva de Permanência de Energia Gerada');
SerieDemEnergia:=Grafico.Series.AddBarSerie('Dem.Energ.',CorDemanda,0,1);
SerieEnergMedia:=Grafico.Series.AddBarSerie('Ener.Gerada',CorEnergia,0,1);
for i := 1 to 12 do
begin
SerieDemEnergia.AddXY(vTempo.Get(i),vDemEnergiaPCR.Get(i),'',CorDemanda);
SerieEnergMedia.AddXY(vTempo.Get(i),vEnergMediaPCR.Get(i),'',CorEnergia);
end;
Grafico.Show;

...
```

Apesar das várias ferramentas nativas do modelo, para analisar os resultados de uma simulação, observou-se que podem ser criadas muitas outras.

As potencialidades de utilização da Rotina de Uso Geral não se encerram aqui. Um usuário inspirado e criativo poderá elaborar tantas ferramentas quantas julgar necessário para monitorar e avaliar um processo de simulação realizado no Propagar MOO.

3.5 SCRIPT GERAL COMO FERRAMENTA DE ANÁLISE INDEPENDENTE DA SIMULAÇÃO

O modelo Propagar MOO permite a execução de scripts independentes da simulação da rede hidrográfica de um projeto. Esses são denominados Scripts Gerais, criados pelo usuário em linguagem Pascal Script.

Os Scripts Gerais podem ser utilizados para várias aplicações, como por exemplo, implementar ferramentas de análise de dados, ou, até mesmo, controlar um processo de simulação. Para demonstrar a aplicabilidade dos scripts gerais, implementou-se uma rotina que faz a leitura de dados de vazão, armazenados em um arquivo, e elabora uma curva de permanência de vazões, apresentando-a em um gráfico.

A rotina elaborada divide-se em quatro etapas principais. Na primeira, realiza-se a leitura dos dados de vazão organizados em um arquivo padrão Excel 4.0, alocando-os em vetores.

Para orientar a leitura dos dados, a rotina necessita que o próprio arquivo de dados possua, em sua estrutura interna, alguma informação referente ao número de dados de vazão armazenados no arquivo. Para demonstrar a organização interna do arquivo de dados usado na aplicação desta rotina, apresenta-se uma parte desses dados na Tabela 3.6.

Tabela 3.6 – Exemplo de estrutura interna do arquivo de vazões usado com o "Script Geral".

Num.Total de Dados	946
Número	**Vazões**
1	25,20
2	22,80
3	20,40
4	18,80
5	24,40
....	....

Na segunda célula da primeira linha encontra-se um valor que representa o número de linhas de dados do arquivo. Na primeira coluna encontra-se um indexador, enquanto que na segunda coluna estão os dados de vazão.

Os comandos da primeira etapa são apresentados a seguir.

```
...

NomeArquivo := 'D:\Diretório\Arquivo.xls';
Planilha := CreateObject(TPlanilha);
Planilha.LoadFromFile(NomeArquivo);
NLV := Trunc(Planilha.GetFloat(1, 2));
NLP := NLV + 2;
NomeSerie := Planilha.GetEntry(2, 2);
vIndiceSerie := CreateObject(TwsDFVec, NLV);
vSerie := CreateObject(TwsDFVec, NLV);
vSerieDecr := CreateObject(TwsDFVec, NLV);
vFreq := CreateObject(TwsDFVec, NLV);
vIndiceSerie := Planilha.ColToVec(1, 3, NLP);
vSerie := Planilha.ColToVec(2, 3, NLP);

...
```

A segunda etapa da rotina é responsável pelos cálculos necessários para a elaboração da curva de permanência de vazões, cujos comandos são apresentados a seguir.

```
...

for i := 1 to NLV do
  vSerieDecr.Set(i, vSerie.Get(i));
vSerieDecr.Sort(false, false);
for i := 1 to NLV do
  vFreq.Set(i, (100*i/NLV));

...
```

Para apresentar os dados calculados na etapa anterior, foram implementadas duas formas de saída. A primeira delas é a apresentação dos dados da curva de permanência na forma de planilha ou tabela, permitindo sua transferência para outros programas.

Outra maneira de apresentar os resultados da curva de permanência de vazões é na forma de um gráfico. Os comandos das etapas que implementam ambas as formas são apresentados a seguir.

Apresentação via tabela:

```
...

Planilha2 := CreateObject(TPlanilha);
Planilha2.Write(1, 1, 'Curva de Permanência da Energia Gerada');
Planilha2.Write(2, 1, 'Projeto ' + Projeto.Nome);
Planilha2.Write(3, 1, 'Num.');
Planilha2.Write(3, 2, NomeSerie);
Planilha2.Write(3, 3, 'Prob.');
Planilha2.WriteVecInCol(vIndiceSerie, 1, 4);
Planilha2.WriteVecInCol(vSerieDecr, 2, 4);
Planilha2.WriteVecInCol(vFreq, 3, 4);
Planilha2.Show;

...
```

Apresentação via gráfico:

```
...

Cor := getColor('TeeColor');
Grafico := CreateObject(TgrGrafico);
Grafico.Chart.SetView3D(False);
Grafico.Chart.Title.Add('Curva de Permanência de ' + NomeSerie);
Serie1 := Grafico.Series.AddLineSerie(NomeSerie, getColor('red'));
for i := 1 to NLV do
  Serie1.AddXY(vFreq.Get(i), vSerieDecr.Get(i), '', Cor);
Grafico.Show;

...
```

CAPÍTULO 4 - APLICAÇÃO.

4.1 O PROPÓSITO DA APLICAÇÃO.

O presente capítulo tem por objetivo demonstrar a aplicação das ferramentas desenvolvidas para planejamento do uso da água e operação de reservatório, simulação da geração de energia e para análise da simulação, implementadas no modelo Propagar MOO, bem como os resultados obtidos.

Para realizar a aplicação das referidas ferramentas, fez-se necessário montar no modelo Propagar uma rede hidrográfica representando uma determinada bacia. Para tal, foi escolhida a Bacia do Rio Paracatu, por ser uma bacia hidrográfica com uma boa quantidade de dados disponíveis, além da existência de outros trabalhos já realizados sobre ela (Pilar, 1998; Lanna, 1998; Viegas Fº, 2000).

Ao ser utilizada como rede hidrográfica básica para a aplicação das ferramentas desenvolvidas, esta bacia hidrográfica não será alvo, nesse estudo, de qualquer análise real sobre as condições de utilização dos seus recursos hídricos. É importante reforçar que o único objetivo de sua utilização como exemplo, foi o de montar um cenário adequado para a aplicação de cada uma das ferramentas elaboradas.

4.2 A BACIA DO RIO PARACATU.

Conforme mencionado anteriormente, o uso da Bacia do Rio Paracatu como base para a construção de uma aplicação do modelo deve-se, principalmente, à quantidade e à organização dos dados disponíveis sobre a bacia.

Entretanto, conforme Viegas Fº (2000), uma aplicação do Propagar tem de ser indicada em função de todo um contexto de uma situação em estudo e que envolva a discussão do atendimento de demandas hídricas ao longo de uma bacia hidrográfica. Dentro dessa perspectiva, a descrição da bacia (adaptada de Viegas Fº, 2000), aqui apresentada, tem por objetivo apresentar o cenário resultante do processo prévio de uso do modelo para construção da rede hidrográfica com seus objetos e suas principais características.

A Bacia do Rio Paracatu (Figura 4.1) localiza-se entre as coordenadas 15º 25’ e 18º 40’ de latitude sul e 45º 03’ e 47º 40’’ de longitude oeste, com uma área de 45.625 km^2, distribuída entre os estados de Minas Gerais (92,3%) e Goiás (4,5%) e o Distrito Federal (3,2%) e integra a sub-bacia do Alto-Médio São Francisco, ficando sua confluência com esse curso de água localizada entre as cidades de Pirapora e São Romão.

Suas principais sub-bacias são, pela margem direita, a do Rio da Prata com 3.750 km2 e a do Rio do Sono com 5.969 km2 e, pela margem esquerda, a do Rio Escuro, com 4.347 km2, a do Rio Preto, com 10.459 km2, e a do Ribeirão Entre-Ribeiros com 3973 km2 (Plano Diretor da Bacia do Rio Paracatu, 1999).

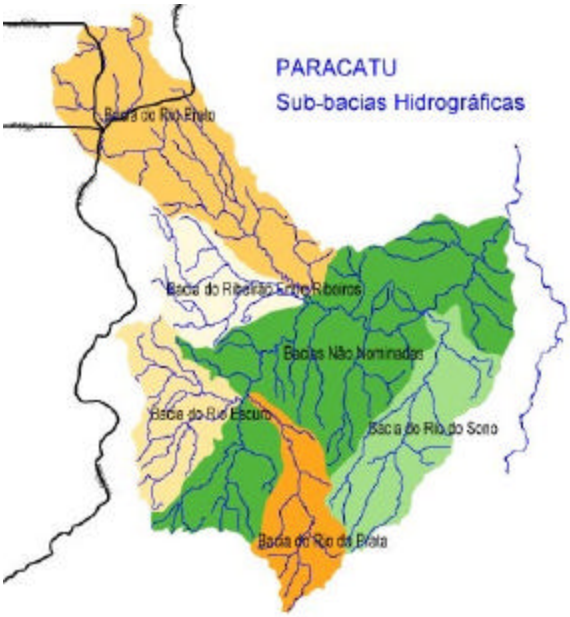


Figura 4. 1 - Bacia do Rio Paracatu e sua sub-bacias.
(Fonte: http://hidricos.mg.gov.br/in-mi.htm)

A precipitação anual, varia de 1600 mm, no limite ocidental, a 1100 mm nos extremos leste e nordeste da bacia. A estação chuvosa compreende os períodos de outubro a abril, quando ocorre cerca de 93% da precipitação anual. A estação seca vai de junho a agosto, quando, então, chove o equivalente a 2% do total anual.

A evaporação anual média é de 1140 mm, com valores mensais variando entre 50 e 60 mm, nos meses de junho a julho, e um máximo de 90 a 130 mm, nos meses de outubro a março. No que se refere à distribuição espacial, a evaporação anual varia entre 1000 mm, nos limites sul e noroeste da bacia, e 1350 mm, no extremo nordeste da mesma.

Os dados de vazão utilizados nas aplicações deste capítulo, já quantificados para cada Ponto Característico, bem como os dados referentes à precipitação e à evaporação sobre o reservatório considerado na simulação, foram obtidos a partir de Lanna (1998) apud Viegas F^{o} (2000). Os dados de vazão de cada PC constituem séries de médias mensais com uma extensão de 55 anos, referentes a um período compreendido entre 1940 e 1994. No caso dos dados de precipitação e evaporação sobre o reservatório, foi utilizada uma única série anual de valores médios mensais.

As atividades econômicas que se desenvolvem na área da bacia hidrográfica e que consubstanciam, com maior significância, as demandas hídricas, são a agricultura - com uma área irrigada existente em torno de 37.600 ha e uma previsão de projetos que a acrescem em cerca de 81.500 ha - e a pecuária - com um rebanho da ordem de 800.000 cabeças (Lanna, 1998, apud Viegas F^{o}, 2000). No que se refere à população urbana da bacia, esta é da ordem de 180.000 habitantes, enquanto que a população rural gira em torno de 140.000 habitantes.

Os dados relativos à formação das demandas hídricas, utilizadas como exemplo neste trabalho, também foram obtidos a partir de Lanna (1998) apud Viegas F^{o} (2000), integrando as tabelas que se seguem. Como todos esses dados já estão relacionados aos Pontos Característicos de interesse dos estudos que foram realizados por ocasião da elaboração do Plano Diretor da Bacia, inseriu-se a Tabela 4.1, que caracteriza e liga os PCs às sub-bacias e às demandas a eles associadas.

Tabela 4.1 - Pontos Característicos, Sub-bacias e Demanda Hídricas.

Ponto Característico	SUB-BACIA	DEMANDAS
PC 01	SB 01	População de Guarda Mor
PC 02	SB 02	Irrigação por aspersão - áreas 6 e 7
PC 03	SB 03	Irrigação por aspersão - área 9
PC 04	SB 04	Irrig. Inundação - área A1, População de Vazante e Lagamar
PC 05	SB 05	População de João Pinheiro e Irrig. Aspersão - áreas 4a, 4b e 5
PC 06	SB 06	Irrig. Inund. - área A2, Pop. de Paracatu e Irrig. asp. áreas 2a e 3
PC 07	SB 07	Expansão Entre Ribeiros e Irrg. Por asp. - áreas 1a, 1b, 2b e 2c
PC 08	SB 08	Barragem do Queimado - geração de energia
PC 09	SB 09	População urbana de Unaí
PC 10	SB 10	Irrigação por aspersão - áreas 8 e 10
PC 11	SB 11	População urbana de Brasilândia
PC 12	SB 12	Irrigação por aspersão - áreas 11, 12 e 13
PC 13	SB 13	Barragem Paracatu - geração de energia
PC 14	SB 14	Irrigação por Inundação - áreas A3 e A4
PC 15	SB 15	Irrig. Inund. - áreas A5 a A10, População de Sta Fé de Minas

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

A Figura 4.2 apresenta um mapa da bacia hidrográfica do rio Paracatu, contendo a localização de quinze pontos de interesse (PCs) definidos dentro da bacia, nomeados de PC_1 a PC_15. Nessa mesma figura, em seu canto superior direito, observa-se o uso do referido mapa como imagem de fundo na área de projeto do Propagar MOO.

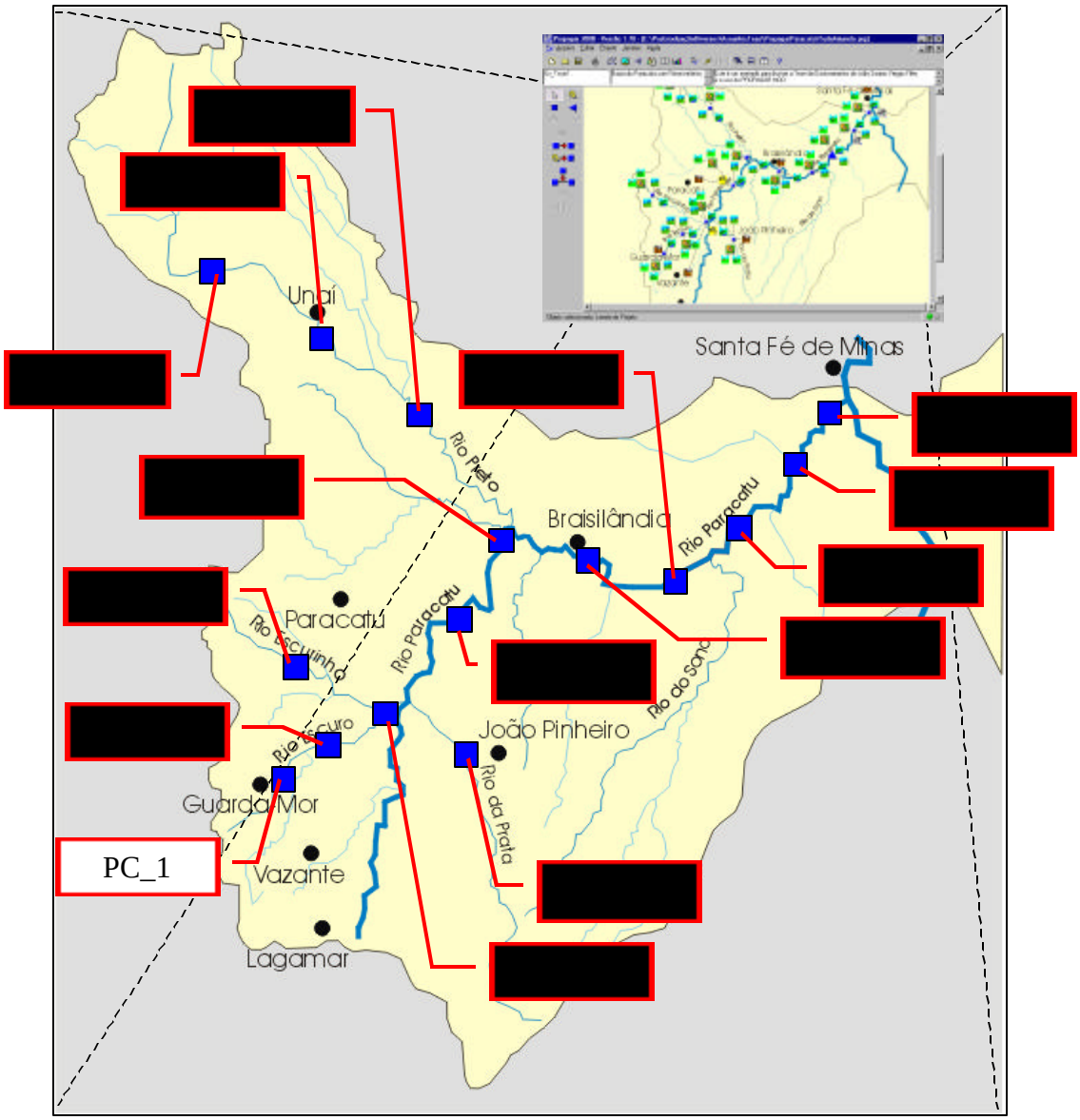


Figura 4.2 - Mapa da Bacia do Rio Paracatu com a localização dos PCs e inserção como figura de fundo da Área de Projeto no PROPAGAR MOO. (Fonte: Viegas Fº, 2000).

A Tabela 4.2 apresenta as demandas difusas relacionadas à população rural e ao rebanho associados a cada PC, indicando, também, os valores referentes aos mesmos.

Tabela 4.2 - Demandas difusas.

Pontos Característicos	Rebanho (cabeças)	População rural (habitantes)	Demanda do rebanho (m^3/s)	Demanda da população rural (m^3/s)	Demanda Difusa (m^3/s)
PC 01	24.936	2.222	0,014	0,002	0,016
PC 02	56.302	3.425	0,032	0,003	0,035
PC 03	88.413	3.831	0,051	0,003	0,054
PC 04	125.153	8.269	0,072	0,007	0,079
PC 05	82.073	7.974	0,048	0,006	0,054
PC 06	53.128	2.560	0,031	0,002	0,033
PC 07	255.590	13.659	0,148	0,011	0,159
PC 08	121.769	66.404	0,070	0,054	0,124
PC 09	94.604	7.272	0,055	0,006	0,061
PC 10	198.463	16.618	0,115	0,013	0,128
PC 11	40.876	3.612	0,024	0,003	0,027
PC 12	48.025	3.976	0,028	0,003	0,031
PC 13	157.785	13.052	0,091	0,011	0,102
PC 14	57.786	7.934	0,034	0,006	0,040
PC 15	24.586	4.928	0,014	0,004	0,018

Fonte: Lanna (1998) apud Viegas F^{o} (2000)..

Nesse trabalho, as vazões ecológicas definidas para cada PC, no plano diretor da bacia hidrográfica do rio Paracatu, foram consideradas como demandas secundárias a serem atendidas em cada ponto. Os valores utilizados para refletir essas demandas, considerada constante ao longo do ano, são apresentados na Tabela 4.3.

Tabela 4.3 - Vazão Ecológica.

Ponto Característico	Vazão Ecológica (m^3/s)	Ponto Característico	Vazão Ecológica (m^3/s)
PC 01	1,39	PC 09	4,92
PC 02	1,86	PC 10	6,75
PC 03	1,07	PC 11	21,45
PC 04	4,78	PC 12	23,04
PC 05	2,72	PC 13	30,14
PC 06	9,54	PC 14	31,76
PC 07	13,99	PC 15	33,09
PC 08	4,27		

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

Os valores das demandas de água dos núcleos urbanos da bacia variam ao longo do ano. Essa variação pode ser observada na Tabela 4.4, onde são apresentadas as demandas urbanas em cada PC. Aqueles que não possuem núcleos urbanos estão com o valor da demanda urbana igual a zero.

Tabela 4.4 - Demandas Urbanas em cada PC, por mês (m3/s).

PC	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
'PC01'	0,028	0,027	0,026	0,026	0,025	0,025	0,027	0,029	0,027	0,025	0,024	0,025
'PC02'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC03'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC04'	0,146	0,157	0,149	0,155	0,153	0,153	0,155	0,167	0,171	0,164	0,141	0,144
'PC05'	0,128	0,125	0,123	0,124	0,120	0,123	0,123	0,129	0,131	0,128	0,121	0,123
'PC06'	0,293	0,308	0,302	0,291	0,299	0,293	0,299	0,291	0,288	0,261	0,282	0,285
'PC07'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
‘PC08’	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC09'	0,197	0,207	0,203	0,195	0,201	0,197	0,201	0,195	0,193	0,176	0,189	0,191
'PC10'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC11'	0,046	0,058	0,054	0,057	0,048	0,051	0,051	0,057	0,057	0,054	0,046	0,045
'PC12'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC13'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC14'	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
'PC15'	0,008	0,010	0,009	0,009	0,008	0,007	0,009	0,009	0,010	0,009	0,008	0,008

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

As demandas localizadas referentes à irrigação dividem-se em dois tipos: as demandas de projetos de irrigação por aspersão e de projetos de irrigação por inundação.

Nos projetos de irrigação por aspersão, as demandas ocorrem entre os meses de março e outubro, conforme demonstra a Tabela 4.5. Os valores tabelados representam as demandas unitárias, ou seja, em cada unidade de área, de cada um dos projetos.

Tabela 4.5 - Demandas unitárias de irrigação por aspersão (l/s/ha).

Mar	Abr	Mai	Jun	Jul	Ago	Set	Out
0,013	0,138	0,163	0,313	0,481	0,631	1,081	0,281

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

Complementando essas informações, apresenta-se na Tabela 4.6 a relação dos projetos de irrigação por aspersão em cada PC, com suas respectivas áreas, que compõem o cenário de demandas utilizado para aplicar as ferramentas desenvolvidas nesse trabalho.

Tabela 4.6 - Áreas relativas às Demandas de Irrigação por Aspersão.

PC	Projetos	Área atual (ha)	Área Proj. (ha)	Área Total (ha)
1	Atual	316,0	-	316,0
2	Atual	2.208,0	-	2.208,0
2	6	-	3.784,0	3.784,0
2	7	-	4.312,0	4.312,0
3	Atual	392,2	-	392,2
3	9	-	2.176,0	2.176,0
4	Atual	1.717,5	-	1.717,5
5	Atual	1.278,0	-	1.278,0
5	4ª	-	5.232,0	5.232,0
5	4b	-	3.248,0	3.248,0
5	5	-	3.968,0	3.968,0
6	Atual	821,0	-	821,0
6	2ª	-	11.536,0	11.536,0
6	3	-	6.016,0	6.016,0
7	Atual	14.476,9	-	14.476,9
7	1ª	-	7.008,0	7.008,0
7	1b	-	512,0	512,0
7	2b	-	6.344,0	6.344,0
7	2c	-	2.816,0	2.816,0
7	Exp. Entre Ribeiros	-	5.500,0	5.500,0
8	Atual	8.153,0	-	8.153,0
9	Atual	586,0	-	586,0
10	Atual	1.345,0	-	1.345,0
10	8	-	4.768,0	4.768,0
10	10	-	2.312,0	2.312,0
11	Atual	1.787,0	-	1.787,0
12	Atual	17,0	-	17,0
12	11	-	5.820,0	5.820,0
12	12	-	3.790,0	3.790,0
12	13	-	2.350,0	2.350,0
13	Atual	4.248,0	-	4.248,0
14	Atual	31,0	-	31,0
15	Atual	220,0	-	220,0
	TOTAIS	37.596,6	81.492,0	119.088,6

Fonte: Lanna (1998) apud Viegas Fº (2000).

De forma similar, as demandas localizadas referentes a projetos de irrigação por inundação têm seus valores unitários variando ao longo do ano, conforme é apresentado na Tabela 4.7. Observa-se, porém, que essa variação ocorre entre os meses de novembro e abril.

Tabela 4.7 - Demandas unitárias de irrigação por inundação (l/s/ha).

Área proj (há)	Jan	Fev	Mar	Abr	...	Nov	Dez
A1 e A2	0,100	2,280	0,733	0,773	...	0,347	1,693
A3 a A10	0,100	2,613	0,987	1,040	...	0,100	2,027

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

As áreas dos projetos de irrigação por inundação, vinculados a cada PC, que foram consideradas na montagem do cenário usado para esse trabalho, são apresentadas na Tabela 4.8.

Tabela 4.8 - Áreas relativas às Demandas de Irrigação por Inundação.

PC	Projetos	Área (ha)
4	A1	1.285
6	A2	1.330
14	A3	335
14	A4	100
15	A5	635
15	A6	620
15	A7	350
15	A8	925
15	A9	290
15	A10	420

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

No ponto característico “PC_8”, foi considerada a existência de um reservatório, denominado Queimado, cuja relação Cota-Área-Volume é apresentada na Tabela 4.9.

Tabela 4.9 - Reservatório Queimado - Relação Cota-Área-Volume.

Cota (m)	Queda (m)	Queda-140 (m)	Área (km^2)	Volume (Hm^3)
780	140	0	0,0	0
790	150	10	2,5	15
800	160	20	6,0	55
810	170	30	23,5	145
820	180	40	26,5	345
830	190	50	41,0	660

Fonte: Lanna (1998) apud Viegas F^{o} (2000).

Esse reservatório estava previsto no Plano Diretor da Bacia do Rio Paracatu, e sua presença na rede hidrográfica é fundamental para criar um cenário que permita a aplicação das ferramentas implementadas nesse trabalho.

4.3 APLICAÇÃO DAS ROTINAS IMPLEMENTADAS

4.3.1 Regra de Defluências Fixas

Para demonstrar a aplicação da regra operacional de Defluências Fixas definidas pelo usuário, optou-se por comparar os resultados do Relatório Geral de Falhas nos PCs, uma das ferramentas de análise de resultados nativas do modelo Propagar MOO, em duas situações: com e sem a utilização da rotina que implementa essa regra operacional. Os relatórios de falhas de ambos os casos são apresentados na Figura 4.3.

Na aplicação dessa regra operacional, foi adotado um conjunto de defluências fixas que eliminasse totalmente as falhas de atendimento das demandas à jusante do reservatório Queimado, mantendo-se um volume de espera, arbitrado em 10% da capacidade máxima do mesmo.

O conjunto de defluências fixas utilizado é demonstrado na Tabela 4.10, que apresenta o conteúdo do arquivo padrão Excel que serve como entrada de dados do usuário para a rotina que implementa essa regra operacional.

Tabela 4.10 - Arquivo de entrada de dados para a regra de Defluências Fixas.

Queimado											
Deflúvios Fixos (m3/s)											
1	1	1	1	1	1	1	5	12	1	1	1
Volumes de Espera (% do Vol. Máximo)											
10	10	10	10	10	10	10	10	10	10	10	10

Na parte superior da Figura 4.3 observa-se que, sem a adoção de qualquer regra operacional, ocorreram falhas terciárias à jusante do reservatório, nos PCs 10, 12 e 13, enquanto que com a aplicação da regra de Defluências Fixas, cujos resultados são apresentados na parte inferior da Figura 4.3, não ocorreram falhas nesses PCs.

As referidas falhas de atendimento das demandas, representam intervalos de tempo, no caso meses, nos quais as demandas não foram totalmente atendidas. Pelos resultados apresentados na Figura 4.3, observa-se que a disponibilidade hídrica dessa bacia hidrográfica, geralmente é suficiente para atender a maioria das demandas impostas pelo cenário previamente definido, sem a necessidade de operar o reservatório.

Falhas no Atendimento das Demandas

	A	B	C	D	E	F	G	H	I	J
1		Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
2	PCs	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
3	PC_5	0	0	0	0	0	0	57	45	38
4	Queimado	0	0	0	0	0	0	0	0	0
5	PC_3	0	0	0	1	0	1	4	4	2
6	PC_1	0	0	0	3	2	3	6	5	5
7	PC_9	0	0	0	0	0	0	0	0	0
8	PC_2	0	0	0	0	0	0	41	33	33
9	PC_4	0	0	0	0	0	0	0	0	0
10	PC_10	0	0	0	0	0	0	12	11	9
11	PC_6	0	0	0	0	0	0	19	19	15
12	PC_7	0	0	0	0	0	0	46	45	38
13	PC_11	0	0	0	0	0	0	0	0	0
14	PC_12	0	0	0	0	0	0	17	15	14
15	PC_13	0	0	0	0	0	0	3	3	3
16	PC_14	0	0	0	0	0	0	0	0	0
17	PC_15	0	0	0	0	0	0	0	0	0
18										

Falhas no Atendimento das Demandas

	A	B	C	D	E	F	G	H	I	J
1		Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
2	PCs	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
3	PC_3	0	0	0	1	0	1	4	4	2
4	PC_1	0	0	0	3	2	3	6	5	5
5	PC_5	0	0	0	0	0	0	57	45	38
6	Queimado	0	0	0	0	0	0	0	0	0
7	PC_2	0	0	0	0	0	0	41	33	33
8	PC_9	0	0	0	0	0	0	0	0	0
9	PC_10	0	0	0	0	0	0	0	0	0
10	PC_4	0	0	0	0	0	0	0	0	0
11	PC_6	0	0	0	0	0	0	19	19	15
12	PC_7	0	0	0	0	0	0	46	45	38
13	PC_11	0	0	0	0	0	0	0	0	0
14	PC_12	0	0	0	0	0	0	0	0	0
15	PC_13	0	0	0	0	0	0	0	0	0
16	PC_14	0	0	0	0	0	0	0	0	0
17	PC_15	0	0	0	0	0	0	0	0	0

Figura 4.3 - Relatório Geral de falhas do PROPAGAR MOO, antes e após a aplicação da regra operacional de Defluências Fixas.

Cabe salientar que a regra de Defluências Fixas não considera a evaporação e a precipitação sobre a superfície do reservatório, além de poder liberar mais água do que o necessário para atender as demandas, visto que não se realiza nenhuma avaliação das mesmas.

4.3.2 Regra Padrão Modificada com Avaliação de Demandas

A utilização dessa rotina para operar um reservatório em uma situação onde as demandas hídricas são atendidas, na maior parte do tempo, pelas afluências hídricas das sub-bacias, principalmente quando o reservatório em questão possui um volume relativamente grande, faz com que as possíveis demandas residuais sejam facilmente atendidas pelo sistema. Devido a não ocorrência de falhas no atendimento das demandas hídricas, a demonstração da aplicação dessa rotina de planejamento ficaria prejudicada.

Para, de fato, demonstrar a potencialidade dessa rotina operacional, optou-se por realizar a sua aplicação em um cenário onde ocorressem falhas de atendimento das demandas hídricas. Assim, foram realizadas duas alterações no cenário original: as demandas de irrigação por aspersão foram dobradas e a capacidade máxima do reservatório foi reduzida de cerca de 628 Hm^3 para 120 Hm^3.

Como o planejamento do uso da água nessa bacia não é o objetivo do presente estudo, o uso de um cenário diferente daquele previamente estabelecido na descrição da bacia não traz prejuízos para os resultados do trabalho.

Inicialmente a rotina criada para implementar a Regra Padrão Modificada foi aplicada com a utilização dos parâmetros mensais listados na Tabela 4.11, que apresenta o conteúdo do arquivo de entrada de dados elaborado pelo usuário do modelo.

De acordo com a primeira linha da referida tabela, definiu-se o conjunto de PCs entre o PC “Queimado” – que corresponde ao PC_8 na Figura 4.2 – e o “PC_15”, como sendo o trecho considerado na avaliação das demandas a serem atendidas pelo reservatório.

Tabela 4.11 - Arquivo de entrada de dados com os parâmetros inicialmente utilizados com a Regra Padrão Modificada.

Queimado	PC_15										
0	0	0	0	0	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10	10	10	10	10

É importante lembrar que, nesse arquivo de entrada de dados, os parâmetros “Alfa” e “Beta”, da Regra Padrão Modificada, não são fornecidos diretamente em unidade de volume (Hm^3), mas na forma de percentual do volume máximo do reservatório. A conversão é realizada pela própria rotina de planejamento, conforme demonstrado no Capítulo 3 do presente trabalho.

Nessa primeira aplicação da rotina, a Regra Padrão Modificada foi utilizada com o seu parâmetro “Beta”, que representa o volume de espera, constante ao longo do ano e arbitrado em 10% da capacidade máxima do reservatório, conforme indicado na terceira linha da Tabela 4.11.

Já o parâmetro “Alfa”, relacionado com o esquema de racionamento estabelecido por essa regra, foi considerado igual a zero durante todo o ano, conforme indicado na segunda linha da Tabela 4.11. Isso significa que, nessa primeira aplicação, foi aplicada a regra padrão original, no que se refere ao esquema de racionamento preventivo.

Para avaliar os resultados da aplicação desse conjunto de parâmetros, a Figura 4.4 apresenta, em sua parte superior, o relatório do conjunto de falhas de atendimento às demandas, ocorridas em cada PC, ao ser usada a rotina que implementa a Regra Padrão Modificada .

Em um segundo momento, aplicou-se a mesma rotina, porém, com o uso de um segundo conjunto de parâmetros, apresentados na Tabela 4.12. Nesse caso, o trecho da rede a ser considerado na avaliação das demandas, foi considerado o mesmo.

É importante lembrar, novamente, que o arquivo de entrada de dados da rotina, cujo conteúdo é apresentado na Tabela 4.12, contém os parâmetros “Alfa” e “Beta” da Regra Padrão Modificada, na forma de percentual do volume máximo do reservatório e não diretamente em unidade de volume.

Esse procedimento foi realizado visando a redução das falhas de atendimento às demandas, através da utilização do esquema de racionamento preventivo estabelecido pela Regra Padrão Modificada.

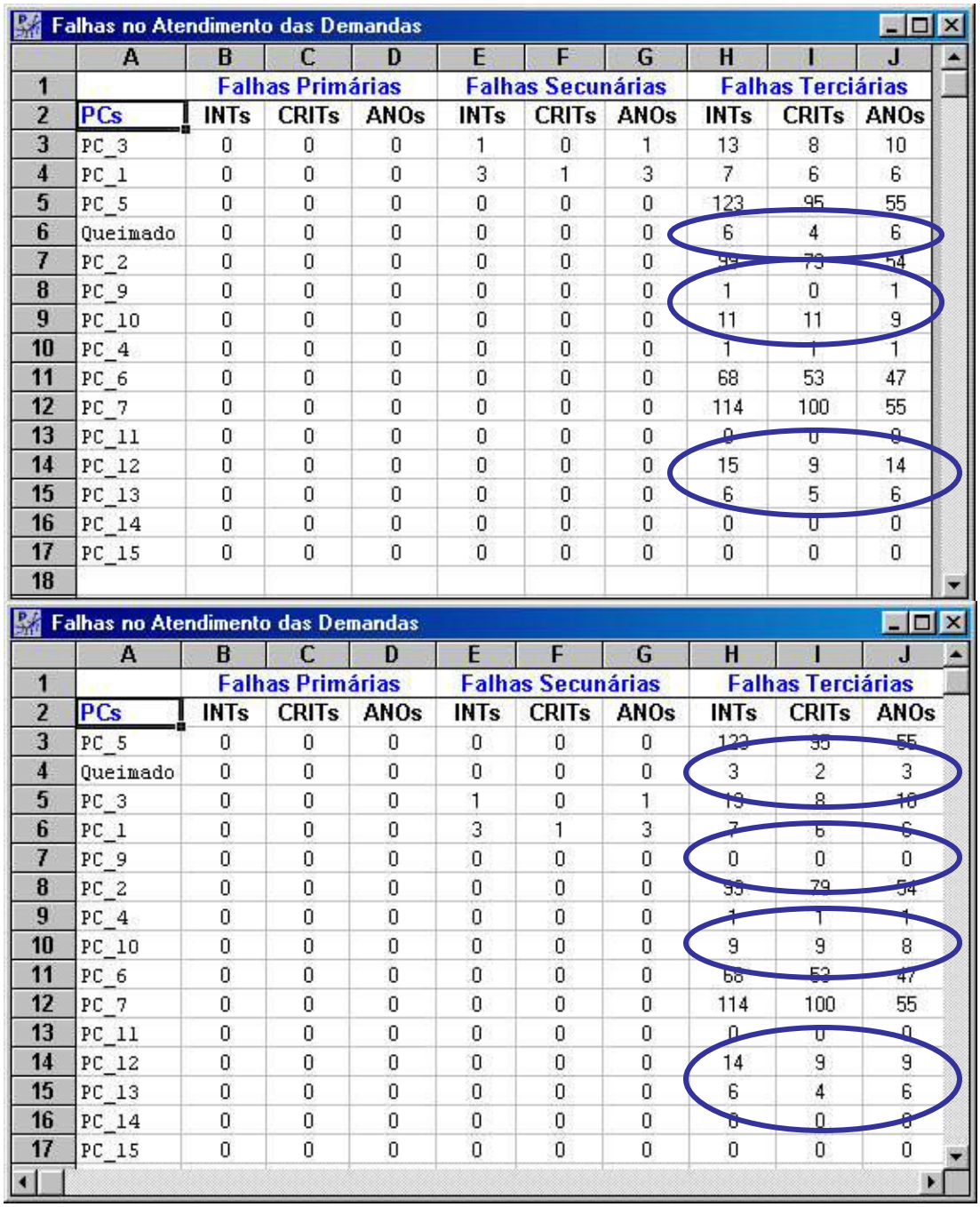
Tabela 4.12 - Arquivo de entrada de dados com os parâmetros posteriormente utilizados com a Regra Padrão Modificada.

Queimado	PC_15										
0	0	0	0	0	40	50	40	0	0	0	0
10	10	10	10	5	0	0	0	0	5	10	10

Para elaborar esse segundo conjunto de parâmetros mensais, foi necessário verificar em que meses do ano ocorriam as falhas de atendimento, ao ser aplicada a regra com o uso do primeiro conjunto de parâmetros. Como a maior parte das falhas ocorria entre os

meses de Agosto e Setembro, procurou-se estabelecer um esquema de racionamento preventivo nos períodos anteriores às falhas, associado ao uso de um menor volume de espera nos meses com falhas e nos meses adjacentes a esses.

Esse processo resultou, após vários testes com diferentes conjuntos de parâmetros, na adoção dos parâmetros apresentados na Tabela 4.12.



Falhas no Atendimento das Demandas

	A	B	C	D	E	F	G	H	I	J
1		Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
2	PCs	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
3	PC_3	0	0	0	1	0	1	13	8	10
4	PC_1	0	0	0	3	1	3	7	6	6
5	PC_5	0	0	0	0	0	0	123	95	55
6	Queimado	0	0	0	0	0	0	6	4	6
7	PC_2	0	0	0	0	0	0	99	79	54
8	PC_9	0	0	0	0	0	0	1	0	1
9	PC_10	0	0	0	0	0	0	11	11	9
10	PC_4	0	0	0	0	0	0	1	1	1
11	PC_6	0	0	0	0	0	0	68	53	47
12	PC_7	0	0	0	0	0	0	114	100	55
13	PC_11	0	0	0	0	0	0	0	0	0
14	PC_12	0	0	0	0	0	0	15	9	14
15	PC_13	0	0	0	0	0	0	6	5	6
16	PC_14	0	0	0	0	0	0	0	0	0
17	PC_15	0	0	0	0	0	0	0	0	0
18										

Falhas no Atendimento das Demandas

	A	B	C	D	E	F	G	H	I	J
1		Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
2	PCs	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
3	PC_5	0	0	0	0	0	0	123	95	55
4	Queimado	0	0	0	0	0	0	3	2	3
5	PC_3	0	0	0	1	0	1	13	8	10
6	PC_1	0	0	0	3	1	3	7	6	6
7	PC_9	0	0	0	0	0	0	0	0	0
8	PC_2	0	0	0	0	0	0	99	79	54
9	PC_4	0	0	0	0	0	0	1	1	1
10	PC_10	0	0	0	0	0	0	9	9	8
11	PC_6	0	0	0	0	0	0	68	53	47
12	PC_7	0	0	0	0	0	0	114	100	55
13	PC_11	0	0	0	0	0	0	0	0	0
14	PC_12	0	0	0	0	0	0	14	9	9
15	PC_13	0	0	0	0	0	0	6	4	6
16	PC_14	0	0	0	0	0	0	0	0	0
17	PC_15	0	0	0	0	0	0	0	0	0

Figura 4.4 - Relatório Geral de Falhas, com a aplicação da Regra Padrão Modificada.

É importante salientar que o conjunto de parâmetros adotado nessa segunda aplicação da rotina, não proporciona, necessariamente, a melhor alternativa possível, no que se refere ao atendimento das demandas hídricas da bacia ou à utilização da água do reservatório. Sua adoção visa mostrar a aplicação da rotina que implementa a Regra Padrão Modificada, bem como demonstrar a possibilidade de redução das falhas no atendimento das demandas hídricas com o uso dessa regra.

O resultado dessa segunda aplicação da Regra Padrão Modificada pode ser observado na Figura 4.4, onde, na sua parte inferior, encontra-se o relatório geral do conjunto de falhas ocorridas em cada PC da rede hidrográfica, ao ser usando o conjunto de parâmetros apresentados na Tabela 4.12.

Observa-se que no PC Queimado existiam 6 falhas no atendimento das demandas terciárias, sendo 4 consideradas críticas (nessa aplicação, uma falha é considerada crítica quando são atendidos menos de 80% do total da demanda). Com o uso do segundo grupo de parâmetros, as falhas nesse PC foram reduzidas para 3, sendo 2 delas consideradas críticas. Observa-se que as falhas dos PCs 9, 10, 12 e 13 também foram reduzidas.

4.3.3 Regra de Volumes-Metas com Avaliação de Demandas

Conforme foi abordado anteriormente, esta regra operacional busca controlar a defluência de um reservatório, atendendo as demandas de um grupo de PCs a jusante, respeitando volumes mínimos e máximos do reservatório ao longo do ano. Ao limitar o volume do reservatório em um valor máximo, a regra operacional cria um volume de espera para controle de cheias, enquanto que, ao estabelecer um volume mínimo para o reservatório, a regra realiza um tipo de racionamento preventivo sobre o uso da água do reservatório, evitando que parte da água seja utilizada no presente, preservando-a para utilização futura.

Para demonstrar a aplicação dessa regra operacional dentro do modelo Propagar MOO, utilizou-se o primeiro cenário de demandas descrito no presente capítulo.

A rotina criada para implementar essa regra, conforme descrito anteriormente, necessita de um conjunto de parâmetros fornecidos pelo usuário, dentre eles os valores mensais dos volumes-metas máximos e mínimos a serem adotados.

A Tabela 4.13 apresenta o conteúdo do arquivo de entrada de dados, demonstrando o conjunto de parâmetros que foi utilizado nessa aplicação.

Tabela 4.13 - Arquivo de entrada de dados com os parâmetros "Volumes-Metas Máximos" e "Volumes-Metas Mínimos" utilizados na aplicação da Regra de Volumes-Metas.

Queimado	PC_15										
Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
80	80	85	90	90	90	100	100	100	100	95	90
30	40	50	60	60	50	10	10	10	10	20	30

Observa-se que na primeira linha do arquivo são informados os nomes dos PCs que delimitam o trecho a ser considerado para a avaliação das demandas, sendo que o primeiro identifica o próprio reservatório a ser operado.

Na terceira linha encontram-se os valores mensais do parâmetro "Volume-Meta Máximo", enquanto que na quarta linha do arquivo estão os valores do parâmetro "Volume-Meta Mínimo". Os valores desses parâmetros estão colocados na forma de percentual da capacidade máxima do reservatório.

Para essa aplicação adotou-se um conjunto de volumes-metas mínimos variável ao longo do ano, sendo maiores nos meses que antecedem o período de maiores demandas, Agosto e Setembro, e menores nesses meses, permitindo o seu melhor atendimento.

O conjunto de parâmetros "Volumes-Metas Máximos" definem qual deve ser o volume de espera a ser mantido no reservatório em cada mês do ano. Em função disso, foram definidos volumes-metas máximos menores durante o período mais chuvoso do ano e valores de Volumes-Metas Máximos mais altos (100%) para o período de maiores demandas, permitindo o uso total da capacidade do reservatório.

Um dos resultados da aplicação da rotina que implementa essa regra operacional, utilizando o conjunto de parâmetros apresentado acima, é o Relatório Geral de Falhas, cujo conteúdo é demonstrado na Tabela 4.14.

Na Tabela 4.14, todos os PCs integrantes do trecho a jusante do reservatório, definido para a aplicação da avaliação de demandas, estão com suas informações nas linhas sombreadas.

Tabela 4.14 – Relatório Geral de Falhas utilizando a Regra de Volumes-Metas.

	Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
PCs	**Ints.**	**Crit.**	**Anos**	**Ints.**	**Crit.**	**Anos**	**Ints.**	**Crit.**	**Anos**
PC_3	0	0	0	1	0	1	4	4	2
PC_1	0	0	0	3	2	3	6	5	5
PC_5	0	0	0	0	0	0	57	45	38
Queimado	0	0	0	0	0	0	0	0	0
PC_2	0	0	0	0	0	0	41	33	33
PC_9	0	0	0	0	0	0	0	0	0
PC_10	0	0	0	0	0	0	0	0	0
PC_4	0	0	0	0	0	0	0	0	0
PC_6	0	0	0	0	0	0	19	19	15
PC_7	0	0	0	0	0	0	46	45	38
PC_11	0	0	0	0	0	0	0	0	0
PC_12	0	0	0	0	0	0	0	0	0
PC_13	0	0	0	0	0	0	0	0	0
PC_14	0	0	0	0	0	0	0	0	0
PC_15	0	0	0	0	0	0	0	0	0

Fazendo uma comparação entre o relatório geral de falhas de atendimento das demandas quando não se utilizam regras operacionais, apresentado na parte superior da Figura 4.3, e o relatório geral de falhas após a aplicação da regra de Volumes-Metas, apresentado na Tabela 4.14, observa-se que este não apresenta a ocorrência de falhas nos PCs das linhas sombreadas. Isso demonstra o correto funcionamento do procedimento de avaliação das demandas para determinar a liberação de água do reservatório.

Com o objetivo de demonstrar o cumprimento da regra de Volumes-Metas, durante o processo de simulação, aplicou-se uma Rotina de Uso Geral denominada “Volumes-Metas”, criada especialmente para ser usada em conjunto com a rotina de planejamento demonstrada nesse item.

A aplicação dessa rotina resulta na construção de um gráfico, apresentado na Figura 4.5, contendo os parâmetros “Volumes-Metas” utilizados durante a simulação, juntamente com as médias dos volumes assumidos pelo reservatório em cada mês do ano.

A utilização de um reservatório de grande volume para atender um cenário com demandas relativamente pequenas, quando comparadas às disponibilidades hídricas naturais da bacia, faz com que o seu volume permaneça normalmente elevado durante o ano, como pode ser observado através dos volumes médios do reservatório demonstrados na Figura 4.5.

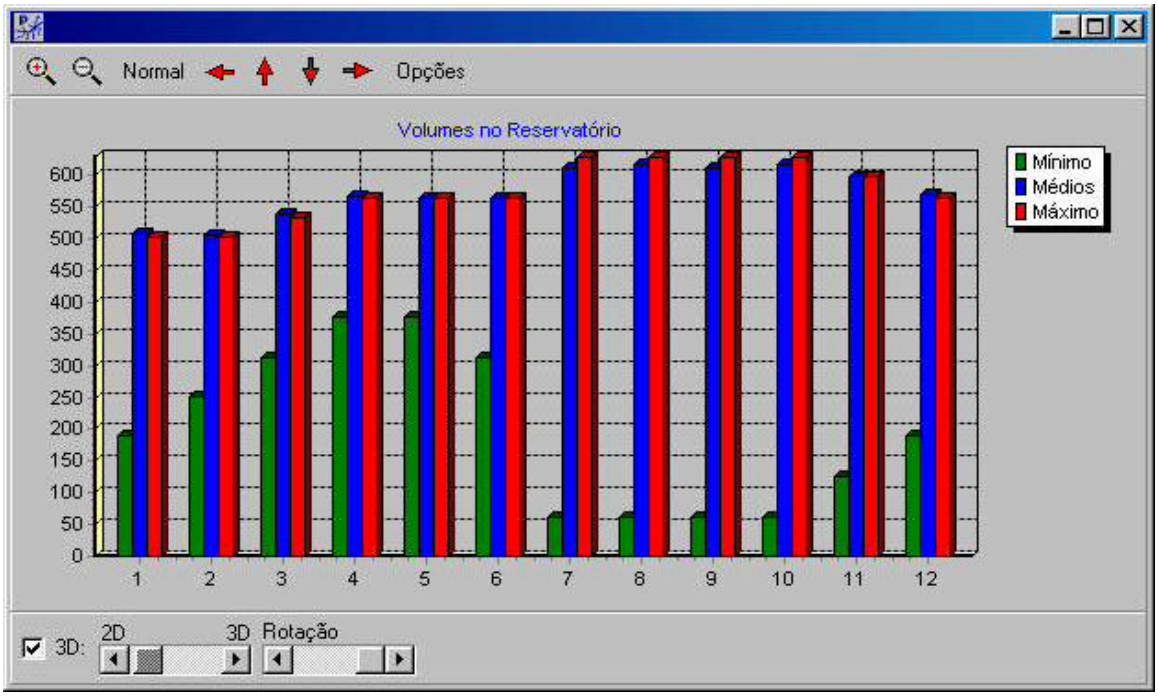


Figura 4.5 – Gráfico dos Volumes-Metas e dos volumes médios do reservatório.

Nessa aplicação, o volume do reservatório permaneceu junto ao limite máximo estabelecido pela regra de volumes-metas, com exceção dos meses de Julho a Outubro, quando ocorrem as maiores demandas na bacia. Na verdade, os limites máximos de volume, estabelecidos através da regra, forçaram o vertimento do reservatório na maior parte do Ano.

Porém, ao comparar os volumes médios assumidos pelo reservatório com as metas de volume máximo estabelecidas, nota-se que os mesmos apresentam valores levemente superiores às metas máximas estabelecidas, nos meses de Dezembro a Abril, o que seria incoerente, à princípio, com a aplicação de uma rotina que estabelece limites máximos para o volume de um reservatório.

Como esses meses se caracterizam pela ocorrência de maiores precipitações na bacia, logo foi levantada a hipótese de que tal diferença estaria ocorrendo em função da metodologia adotada na criação dessas rotinas de planejamento, de não considerar a precipitação e a evaporação sobre a superfície do reservatório.

Mesmo afastando-se da proposta original, seria importante verificar tal hipótese para que as rotinas utilizadas fossem melhor avaliadas, além de estimar o erro que resulta da aplicação dessa metodologia.

Para tal, alterou-se a rotina de planejamento de Volumes-Metas para que essa viesse a contemplar a precipitação e a evaporação sobre o reservatório. Os resultados obtidos por essa alteração são apresentados na Figura 4.6.

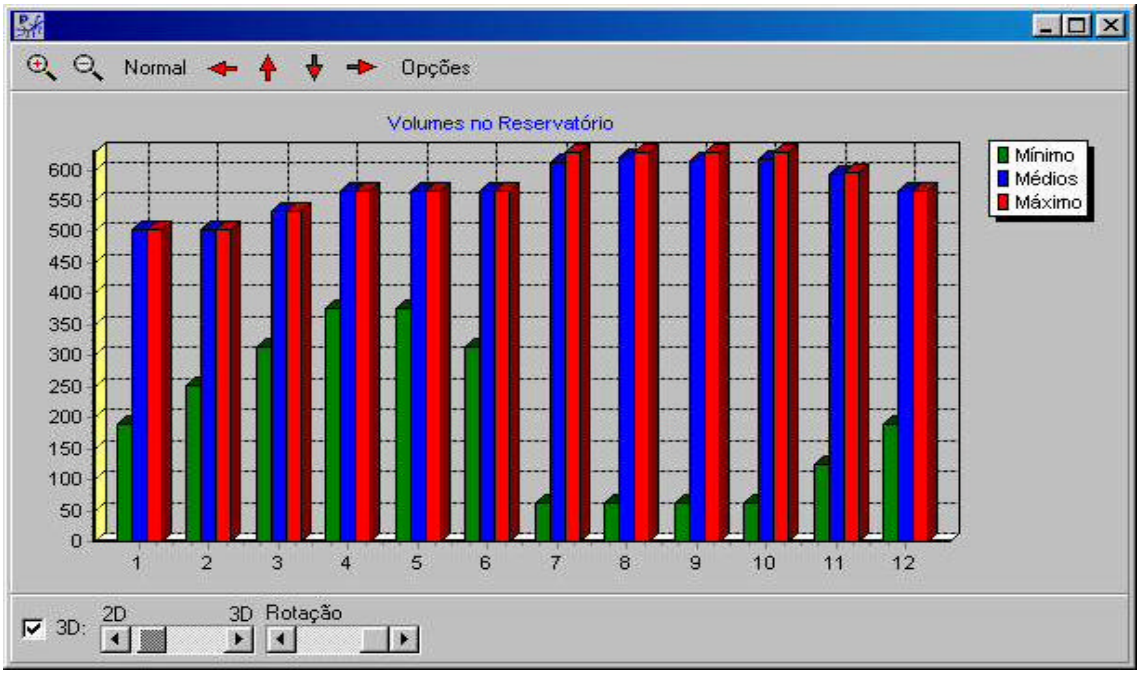


Figura 4.6 – Gráfico dos Volumes-Metas e dos volumes médios do reservatório, considerando a evaporação e a precipitação sobre a superfície do reservatório.

Observa-se na Figura 4.6 que os valores das médias mensais dos volumes assumidos pelo reservatório ao longo da simulação, em nenhum período ultrapassam as metas de volume máximo estabelecidas pela regra, o que comprova a hipótese anteriormente levantada.

Para considerar a evaporação e a precipitação sobre o reservatório na avaliação da disponibilidade hídrica do mesmo, foi necessário implementar dentro da rotina original um conjunto de comandos que estabelecem iterações controladas no cálculo, pois a influência da precipitação e da evaporação sobre a disponibilidade hídrica varia de acordo com a área da superfície do reservatório, e esta, de acordo com a variação de volume ocorrida em cada intervalo de tempo.

As alterações sobre a rotina original de planejamento de Volumes-Metas, para permitir que essa contemplasse a precipitação e a evaporação sobre o reservatório, foram realizadas basicamente na etapa de aplicação da regra operacional e são abordadas a seguir.

A nova etapa de aplicação da regra operacional passou a contar com o comando *While...do*, que cria e controla um laço, dentro do qual encontra-se a etapa original de aplicação da regra operacional, além de outros comandos. O conjunto de comandos em Pascal Script que precede a antiga aplicação da regra de Volumes-Metas é apresentada a seguir.

```
...
CicloEvapo := 0;
VolFinal := VolInicInterv;
while CicloEvapo <= 10 do
  begin
  VolMedio := (VolInicInterv + VolFinal)/2;
  AreaMedia := PCR.CalculaArea(VolMedio);
  AfluAux := 0.001 * AreaMedia * (PCR.ObtemPrecipitacaoUnitaria(dt)
             - PCR.ObtemEvaporacaoUnitaria(dt));
  VolAtual := VolInicInterv + AfluHm3 + VazaoPCsMontanteRes + AfluAux
              - DemReservatorio;
...
  {Conjunto de comandos que aplicam a regra operacional de forma similar à
   rotina original de Volumes-Metas}
...
  VolAtual := VolAtual - PCR.m3_Hm3_Intervalo(L);
  Difer := ABS(VolFinal – VolAtual);
  if Difer <= (0.01 * PCR.VolumeMaximo) then CicloEvapo := 11
  else CicloEvapo := CicloEvapo + 1;
  VolFinal := VolAtual;
  end;
PCR.AtribuiDefluvioPlanejado(dt, L);
...
```

O laço realiza um controle, utilizando a variável auxiliar *cicloevapo*, para que o ciclo de cálculo não ultrapasse 10 iterações. No princípio do laço, calcula-se o volume médio, a área média, a afluência auxiliar (precipitação e evaporação sobre o reservatório, que depende da área média) e o volume atual do reservatório, que integra todas as contribuições e retiradas de água do reservatório, sendo utilizado na aplicação da regra operacional propriamente dita.

Na aplicação da regra operacional define-se qual a defluência a ser planejada para o reservatório, em função do seu volume atual. Após isso, desconta-se do próprio volume atual, o valor do deflúvio definido, calculando-se o valor absoluto da diferença entre esse novo volume atual e o conteúdo da variável *volfinal*. Se essa diferença for menor que 1% do volume máximo do reservatório, o ciclo de iterações é interrompido.

Ao final de cada ciclo, atualiza-se a variável *volfinal* com o valor do volume atual calculado, preparando o novo ciclo de cálculo.

Para auxiliar em aplicações futuras, bem como dirimir possíveis dúvidas, essa rotina operacional é apresentada integralmente no Anexo 1 do presente trabalho.

4.3.4 Regra de Zoneamento de Reservatório com Avaliação de Demandas

Conforme abordado anteriormente, essa regra operacional realiza uma divisão do volume do reservatório em zonas, a partir de volumes pré-estabelecidos. A definição de quais demandas poderão ser atendidas, entre primárias, secundárias e terciárias, será de acordo com a zona na qual se encontra o volume do reservatório no início do intervalo de simulação.

Visando demonstrar sua aplicação da rotina que implementa essa regra, realizou-se uma comparação entre as falhas no atendimento das demandas, com e sem a utilização da regra de zoneamento de reservatório. Para isso, foi necessário realizar alterações no cenário original de disponibilidades hídricas naturais e de demandas da bacia, de forma que fosse criado um outro cenário onde ocorressem falhas de atendimento das demandas hídricas. As alterações realizadas no cenário original foram as seguintes: as vazões afluentes de cada sub-bacia foram reduzidas em 40%, gerando uma situação de menor disponibilidade hídrica; as vazões ecológicas, definidas como demandas secundárias, foram dobradas; a capacidade máxima do reservatório foi reduzida para 100 Hm^3, com volume mínimo de 5 Hm^3.

Diante disso, é importante reafirmar que o planejamento do uso da água nessa bacia não é objetivo desse estudo, logo o uso de um cenário diferente daquele previamente estabelecido na descrição da bacia não irá prejudicar os resultados do trabalho.

Inicialmente a rotina que implementa o zoneamento de reservatório foi aplicada com a utilização dos parâmetros listados na Tabela 4.15, introduzidos na simulação através de um arquivo padrão Excel 4.0, cuja leitura é realizada pela própria rotina.

Tabela 4.15 – Arquivo de entrada de dados com os limites superiores das zonas do reservatório, simulando a não aplicação da regra de zoneamento (em % do volume máximo).

Queimado	PC_15										
5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5
100	100	100	100	100	100	100	100	100	100	100	100

A utilização dos parâmetros apresentados na Tabela 4.15 simula uma situação onde o reservatório esteja presente e seja realizada a avaliação das demandas, porém, sem a aplicação da regra de zoneamento de reservatório. Isso ocorre porque todo o volume útil do reservatório poderá ser utilizado para atender quaisquer tipos de demanda, pois os volumes mensais que limitam as zonas 1 e 2, respectivamente localizados nas linhas 2 e 3 da tabela, são iguais ao volume mínimo do reservatório durante todo o ano.

Dessa forma, foi possível avaliar a quantidade de falhas no atendimento das demandas, sem a aplicação da regra de zoneamento, apesar do reservatório estar presente e ser utilizado para atender demandas. O resultado da aplicação desse primeiro conjunto de parâmetros é apresentado na Tabela 4.16.

Tabela 4.16 – Relatório Geral de Falhas, sem aplicar a Regra de Zoneamento de Reservatório.

	Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
PCs	**INTs**	**CRITs**	**ANOs**	**INTs**	**CRITs**	**ANOs**	**INTs**	**CRITs**	**ANOs**
PC_3	0	0	0	19	18	11	40	39	21
PC_1	0	0	0	144	114	48	155	155	50
PC_5	0	0	0	30	24	14	159	152	55
Queimado	0	0	0	20	18	11	28	28	17
PC_2	0	0	0	9	6	6	126	117	55
PC_9	0	0	0	29	29	17	28	28	17
PC_10	0	0	0	29	29	17	29	29	18
PC_4	0	0	0	15	9	9	21	20	13
PC_6	0	0	0	30	20	13	109	102	49
PC_7	0	0	0	33	22	14	148	144	55
PC_11	0	0	0	30	29	17	29	29	17
PC_12	0	0	0	30	29	17	37	36	22
PC_13	0	0	0	36	30	19	41	41	23
PC_14	0	0	0	37	30	21	38	38	21
PC_15	0	0	0	37	31	21	38	38	21

A identificação do PC reservatório e dos PCs localizados a jusante dele, bem como as falhas secundárias e terciárias dos mesmos, são apresentados nas linhas sombreadas.

Na seqüência, a mesma rotina foi aplicada, porém, com outro conjunto de parâmetros, apresentados na Tabela 4.17.

Tabela 4.17 – Arquivo de entrada de dados com os limites superiores das zonas do reservatório, para a aplicação da regra de zoneamento de reservatório (em % do volume máximo).

Queimado	PC_15										
8	8	8	8	8	8	8	8	5	5	8	8
40	40	40	40	60	60	60	50	40	40	40	40
100	100	100	100	100	100	100	100	100	100	100	100

Para elaborar esse segundo conjunto de parâmetros mensais, verificou-se em que meses do ano ocorriam as falhas de atendimento, quando da aplicação da rotina de zoneamento e do uso do primeiro conjunto de parâmetros.

Posteriormente, buscou-se um conjunto de parâmetros que implementasse um zoneamento no reservatório, de forma que ocorresse um racionamento preventivo das

demandas terciárias nos meses que antecediam as falhas. Com isso, a água permaneceria no reservatório para utilização futura e atenderia melhor as demandas secundárias.

O resultado da aplicação do segundo conjunto de parâmetros é apresentado na Tabela 4.18, através do Relatório Geral de Falhas.

Tabela 4.18 – Relatório Geral de Falhas, aplicando a Regra de Zoneamento de Reservatório.

	Falhas Primárias			Falhas Secundárias			Falhas Terciárias		
PCs	INTs	CRITs	ANOs	INTs	CRITs	ANOs	INTs	CRITs	ANOs
PC_3	0	0	0	19	18	11	40	39	21
PC_1	0	0	0	144	114	48	155	155	50
PC_5	0	0	0	30	24	14	159	152	55
Queimado	0	0	0	19	17	11	27	27	17
PC_2	0	0	0	9	6	6	126	117	55
PC_9	0	0	0	29	29	17	53	53	29
PC_10	0	0	0	29	29	17	53	53	29
PC_4	0	0	0	15	9	9	21	20	13
PC_6	0	0	0	30	20	13	109	102	49
PC_7	0	0	0	33	22	14	148	144	55
PC_11	0	0	0	29	29	17	53	53	29
PC_12	0	0	0	29	29	17	56	56	29
PC_13	0	0	0	33	29	18	60	60	29
PC_14	0	0	0	33	30	19	74	74	31
PC_15	0	0	0	33	30	19	74	74	31

Ao realizar a comparação entre as tabelas 4.16 e 4.18, observa-se que no PC Queimado (reservatório), as falhas de atendimento das demandas secundárias reduziram de 20 para 19, sendo que as falhas críticas também reduziram, com a aplicação da regra de zoneamento de reservatório, enquanto que nos PCs 11, 12, 13, 14 e 15, ocorreram reduções no número de falhas de atendimento às demandas.

Como a regra aplica um racionamento preventivo da água para o atendimento das demandas terciárias, estas apresentaram um maior número de falhas, como já era esperado.

As falhas no atendimento das demandas secundárias, observadas na simulação, ocorreram em vários meses, ao longo da série de dados históricos utilizada, sempre nos períodos de menor afluência hídrica das sub-bacias e quando o volume de água do reservatório reduzia para valores abaixo do limite da zona 1. Com o objetivo de demonstrar essa redução de volume, apresenta-se na Figura 4.7 A variação de volume do reservatório ao longo do ano de 1960, quando ocorrem falhas nos meses de Setembro a Novembro da série de dados utilizada na simulação.

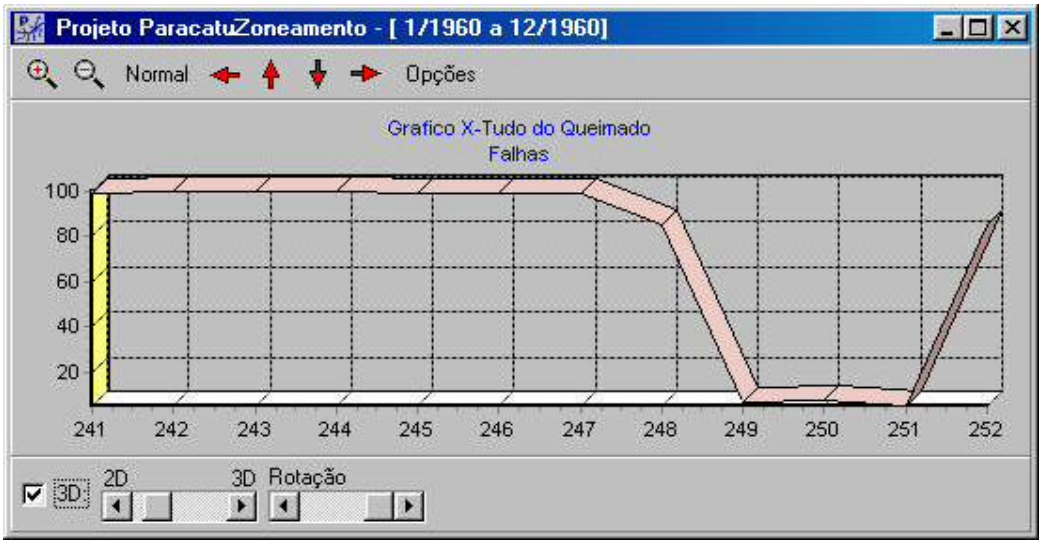


Figura 4.7 – Volume do reservatório Queimado no ano de 1960 (em Hm3).

Como essa bacia hidrográfica apresenta pequenas demandas primárias e uma grande disponibilidade hídrica, não foi possível, mesmo com alterações de cenário, provocar falhas no atendimento das demandas primárias e assim demonstrar a influência do zoneamento sobre o atendimento dessas demandas.

4.3.5 Cálculo Padrão de Energia Gerada

Esse item visa demonstrar o uso do cálculo padrão, implementado no balanço hídrico do modelo Propagar MOO, que faz a estimativa da energia gerada em um ponto da rede com usina hidrelétrica. Para isso foi considerada na simulação a existência de uma usina hidrelétrica no reservatório Queimado.

Quando da obtenção dos dados relativos ao cenário utilizado nesse trabalho, a referida usina encontrava-se em fase de análise da sua viabilidade econômica, estando, hoje, em fase de construção.

Uma vez que a usina possui um reservatório de regularização, para que o modelo possa simular a geração de energia é preciso que seja planejada uma defluência de água desse reservatório. Para aplicar o cálculo padrão de energia gerada utilizou-se a rotina de planejamento de Defluências Fixas, já apresentada anteriormente.

Fazendo uma rápida revisão, a referida rotina de planejamento permite que o usuário determine quais as defluências mensais previstas para o reservatório, através de um arquivo padrão Excel 4.0. O conteúdo do arquivo utilizado nessa aplicação, contendo as defluências adotadas, é apresentado na Tabela 4.19.

Tabela 4.19 – Arquivo de entrada de dados com os parâmetros de defluência fixa mensal utilizados na aplicação do cálculo padrão da energia gerada no reservatório.

Queimado											
Deflúvio	(m^3/s)										
40	40	30	30	30	30	30	25	25	30	40	40

Para o preenchimento da janela de entrada de dados referentes à geração de energia, apresentada no Capítulo 3 desse trabalho, foram adotados os seguintes valores:

- Rendimento do sistema de adução = 0,97
- Rendimento das turbinas = 0,90
- Rendimento dos geradores = 0,93
- Máxima vazão turbinável = 70 m^3/s
- Cota de jusante da usina = 638 m

Para apresentação dos resultados obtidos no cálculo de energia, com a utilização dos dados listados acima, existem algumas ferramentas nativas do modelo Propagar MOO. Uma dessas ferramentas é o gráfico da energia gerada, para o qual pode-se definir o período da simulação a ser apresentado. Na Figura 4.8 apresenta-se o gráfico da energia gerada na usina do PC Queimado, durante todo o período de dados da simulação.

Observa-se na Figura 4.8 uma variação cíclica da energia gerada durante a simulação. Nota-se que os valores variam de 25 a 45 GWh em cada mês, aproximadamente, e a causa é a adoção de diferentes vazões a serem liberadas do reservatório ao longo do ano.

Também se pode observar a presença de períodos onde a energia gerada assume valores bem inferiores à média. Essa redução ocasional no valor mensal da energia gerada é causada pela redução da disponibilidade hídrica do reservatório, em momentos de menor afluência hídrica ao reservatório.

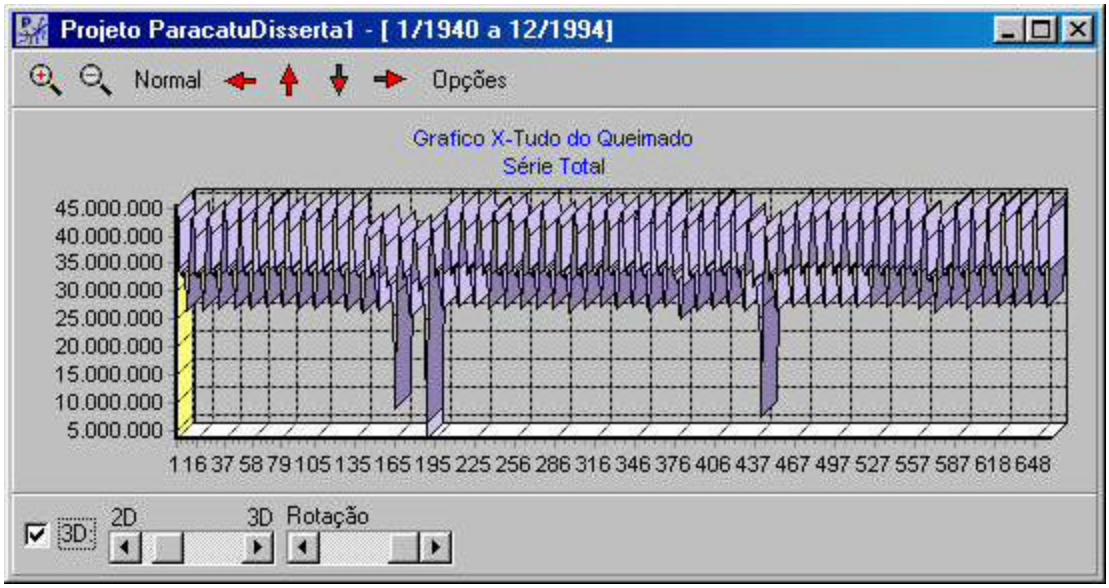


Figura 4.8 – Gráfico da Energia gerada pela usina do reservatório Queimado, durante toda a série de dados hidrológicos (em KWh/mês).

A Figura 4.9, por apresentar o gráfico da energia gerada durante um período de tempo reduzido, permite uma melhor visualização, tanto da variação cíclica, como da redução ocasional da energia gerada.

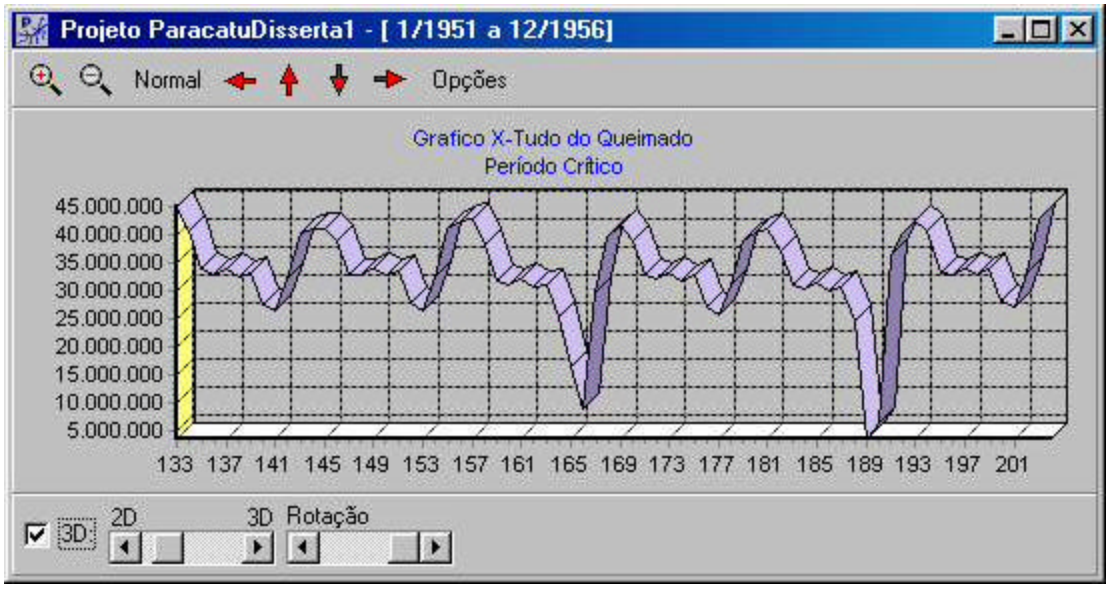


Figura 4.9 – Gráfico da Energia gerada pela usina do reservatório Queimado, durante o período de Jan/1951 a Dez/1956 da série hidrológica (em KWh/mês).

A Figura 4.9 mostra que nos intervalos de simulação identificados pelos Nºs 165 e 189 ocorreram as maiores reduções na geração de energia. Esses intervalos de simulação estão relacionados com o período de menor afluência hídrica ao reservatório, nos meses de Julho a Outubro.

4.3.6 Rotina para Cálculo da Energia Gerada

O modelo Propagar MOO permite que o usuário utilize, para o cálculo da energia gerada, uma metodologia diferente daquela adotada no cálculo padrão do modelo. Conforme abordado no capítulo anterior, isso pode ser feito através da criação de uma rotina em Pascal Script, a ser executada pelo modelo como uma Rotina de Cálculo de Energia.

No Capítulo 3 do presente trabalho, apresenta-se a metodologia da rotina de cálculo de energia criada para demonstrar o uso desse tipo de rotina, sendo os resultados de sua aplicação, demonstrados nesse item.

A rotina implementada faz a leitura de um arquivo, padrão Excel 4.0, contendo os dados de uma curva que relaciona o rendimento das turbinas, a ser adotado no cálculo da energia gerada, com a vazão turbinada, na forma de percentual da vazão máxima turbinável. O conteúdo desse arquivo de dados é apresentado na Tabela 4.20.

Tabela 4.20 – Conteúdo do arquivo de entrada de dados para a rotina de Cálculo de Energia.

PC_8	11
Q/Qmax (%)	Rendimento
8	2
10	10
20	54
30	69
40	77
50	83
60	86
70	88
80	89
90	89
100	87

Os demais dados utilizados para realizar o cálculo de energia através dessa rotina foram mantidos idênticos aos usados no cálculo demonstrado no item anterior, como, por exemplo, o rendimento dos geradores, ou a máxima vazão turbinável.

Mesmo com a utilização de uma rotina de cálculo de energia, é necessário realizar a liberação de água do reservatório através de uma rotina de planejamento. Para isso utilizou-se a mesma rotina usada na demonstração do cálculo padrão da energia gerada, sendo adotado o mesmo arquivo de defluências fixas ao longo do ano.

Para demonstrar os resultados obtidos com a aplicação da rotina de cálculo de energia, adotou-se a mesma forma utilizada na apresentação dos resultados do cálculo padrão da geração de energia, ou seja, um gráfico contendo a energia gerada em cada intervalo de tempo. Porém, optou-se por apresentar somente o período de Janeiro de 1951 a Dezembro de 1956, para facilitar a identificação dos valores, o que é feito através da Figura 4.10.

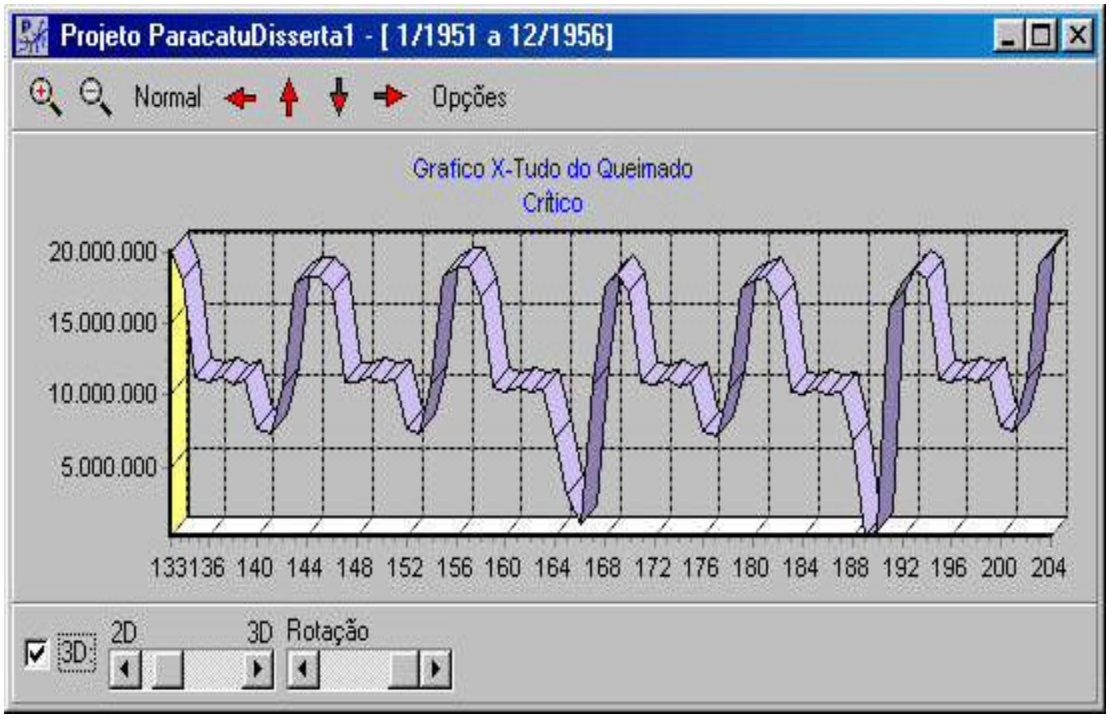


Figura 4.10 – Gráfico da Energia gerada pela usina do reservatório Queimado, usando a Rotina de Cálculo de Energia, durante o período de Jan/1951 a Dez/1956 (em KWh/mês).

Observa-se que a forma básica da curva da energia gerada, apresentada acima, é a mesma daquela apresentada na Figura 4.9, pois se trata da mesma série de afluências ao reservatório e do mesmo período de resultados graficados.

Porém, os valores resultantes do cálculo da energia gerada são inferiores aos obtidos pelo cálculo padrão, pois naquele caso o rendimento das turbinas foi considerado constante e igual a 90%, enquanto que na rotina de cálculo de energia em Pascal Script, o rendimento das turbinas assume valores diferentes, conforme a vazão turbinada.

Assim demonstra-se como o usuário pode alterar qualquer um dos parâmetros utilizados no cálculo ou, até mesmo, a própria metodologia de cálculo da energia gerada em um PC com usina hidrelétrica.

4.3.7 Rotina de Planejamento com Avaliação das Demandas de Energia e Rotina de Uso Geral para Cálculo da Energia Média Gerada

Nesse item será demonstrada a aplicação da rotina de planejamento que realiza a determinação da quantidade de água a ser liberada ou turbinada pela usina, a cada intervalo de simulação, através da avaliação das demandas de energia informadas pelo usuário.

Na construção da rede hidrográfica, na interface gráfica do Propagar, o usuário deve informar o nome de um arquivo padrão texto (.txt), no qual estão os valores das demandas de energia para mês do ano, em KWh. O nome do referido arquivo deve ser introduzido no campo denominado “Arquivo com a Curva de Demanda Energética”, existente na janela de diálogo sobre dados de energia de um PC, ambos abordados anteriormente.

Para a presente aplicação, foi criado um arquivo de texto contendo as demandas mensais de energia, em KWh, conforme apresentado a seguir:

```
50000000
50000000
45000000
40000000
35000000
30000000
25000000
20000000
20000000
30000000
40000000
50000000
```

Essa seqüência de valores foi determinada de forma casual, apenas para ilustrar a aplicação dessa rotina de planejamento.

O uso dessa rotina faz com que a quantidade de energia gerada na usina, em cada intervalo de simulação, seja, em média, muito próxima da demanda, definida pelo usuário. Uma redução cíclica da disponibilidade hídrica do reservatório pode aumentar de maneira significativa a diferença entre a energia média gerada em cada mês e a respectiva demanda desse mês.

Para demonstrar os resultados obtidos com a aplicação da rotina de planejamento que avalia as demandas mensais de energia, implementou-se uma rotina de uso geral para gerar um gráfico que apresenta a demanda energética de cada mês, juntamente com a energia média gerada ao longo do ano. Esse gráfico é apresentado na Figura 4.11.

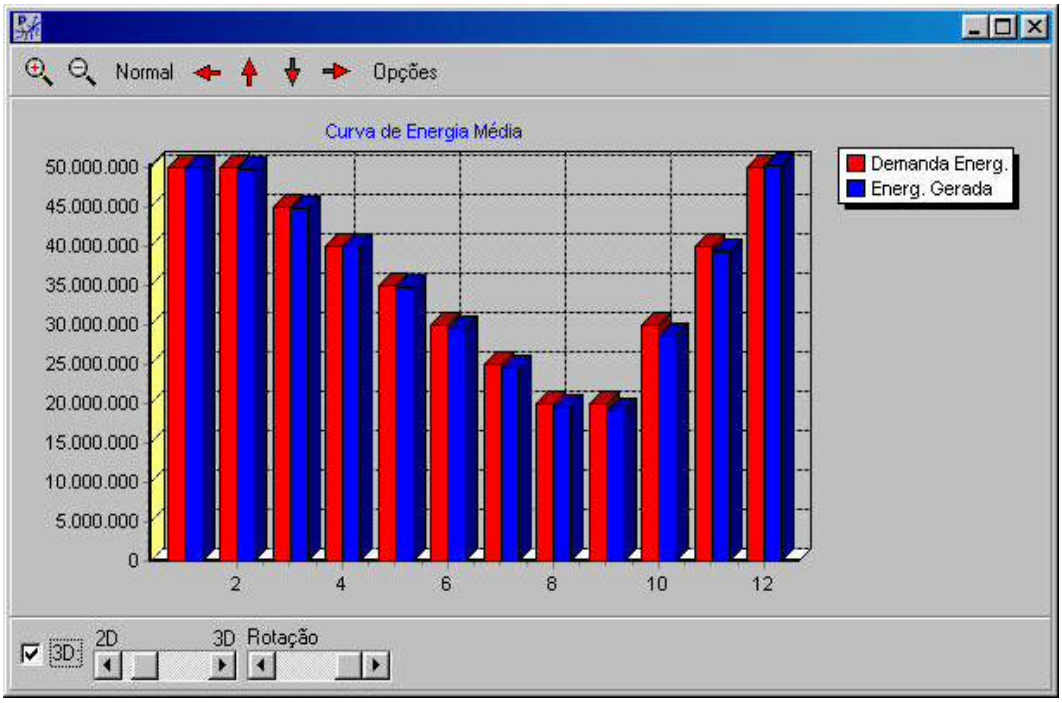


Figura 4.11 – Gráfico da demanda mensal de energia e da energia média gerada pela usina do reservatório Queimado (em KWh/mês).

As pequenas diferenças entre a energia média gerada e a demanda energética, que ocorrem em determinados meses, são causadas pelo fato de que na rotina de planejamento, para fazer a determinação da queda hidráulica do reservatório em cada intervalo de tempo, utiliza-se o volume do reservatório no início do intervalo de simulação. O processo mais preciso seria calcular a média entre os volumes inicial e final do reservatório, em cada intervalo de tempo, o que tornaria necessário um processo iterativo de cálculo, acrescido de balanço hídrico.

Porém, para manter a simplicidade dessa rotina de planejamento, optou-se por realizar um cálculo menos preciso. Mesmo com essa simplificação, as diferenças foram muito pequenas.

4.3.8 Rotina de Uso Geral para Cálculo e Apresentação da Curva de Permanência de Energia Gerada

Para a aplicação dessa rotina, novamente considera-se a presença de uma usina hidrelétrica no reservatório queimado. Para operar o reservatório e determinar a liberação de água, utilizou-se a rotina de planejamento que determina defluências fixas para cada mês do ano, abordada anteriormente.

Como resultado da aplicação dessa rotina de uso geral, obtém-se a curva de permanência apresentada na Figura 4.12.

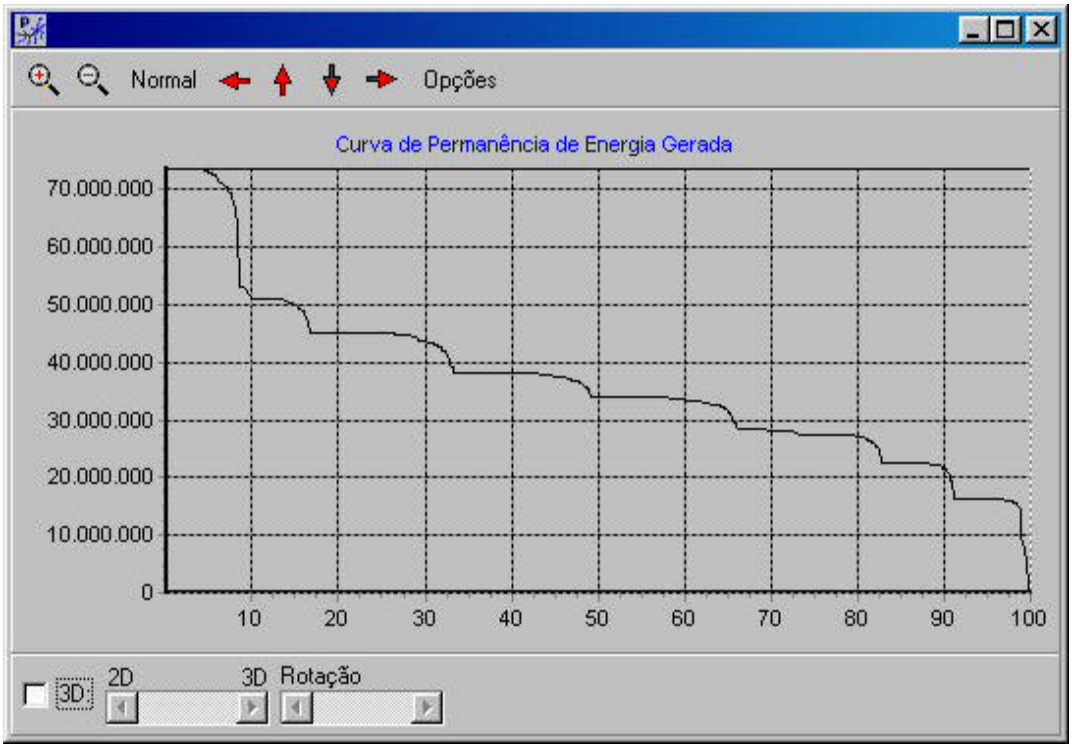


Figura 4.12 – Curva de Permanência da Energia Gerada no PC Queimado (KWh/mês x %).

Essa rotina de uso geral também gera uma tabela com os dados que originaram a curva apresentada acima, de forma que estes podem ser transferidos para uma planilha eletrônica. A Tabela 4.21 apresenta parte dos resultados gerados pela aplicação dessa rotina de uso geral.

Tabela 4.21 – Conteúdo do arquivo de entrada de dados para a rotina de Cálculo de Energia.

Curva de Permanência da Energia Gerada		
Dt	**Energia**	**Probabilidade**
1	73.567.480,00	0,15152
2	73.567.480,00	0,30303
3	73.567.480,00	0,45455
4	73.567.480,00	0,60606
. . .	. . .	. . .
171	45.007.608,00	25,90909
172	44.988.248,00	26,06061
173	44.971.964,00	26,21212
174	44.968.872,00	26,36364
175	44.934.932,00	26,51515
176	44.901.016,00	26,66667
177	44.895.864,00	26,81818
178	44.871.328,00	26,96970
179	44.787.796,00	27,12121
180	44.784.432,00	27,27273
. . .	. . .	. . .
423	32.479.794,00	64,09091
424	32.384.270,00	64,24242
425	32.216.292,00	64,39394
426	32.112.756,00	64,54545
427	32.002.322,00	64,69697
428	31.942.908,00	64,84848
429	31.670.722,00	65,00000
430	31.270.154,00	65,15152
431	31.174.994,00	65,30303
432	30.544.640,00	65,45455
433	30.308.276,00	65,60606
434	29.599.608,00	65,75758
435	29.481.758,00	65,90909
436	29.176.444,00	66,06061
. . .	. . .	. . .
658	6.184.922,00	99,69697
659	807.509,25	99,84848
660	-	100,00000

4.3.9 Rotina “Script Geral” para a Elaboração de uma Curva de Permanência de Vazões

Conforme abordado anteriormente, o Propagar MOO permite que sejam criadas rotinas em Pascal Script para serem executadas de forma independente da simulação do modelo. Para isso, as rotinas criadas com essa finalidade devem ser incluídas na relação de “Scripts Gerais” de um determinado projeto.

Embora os Scripts Gerais estejam ligados a um determinado projeto, devido à estrutura do modelo, seu processamento é totalmente independente do mesmo. Dessa forma, uma determinada rotina Script Geral pode acessar e processar dados e informações totalmente desvinculados do projeto ao qual estiver ligado.

Para demonstrar essa característica, bem como demonstrar a aplicação de um “Script Geral”, utilizou-se uma rotina que calcula e apresenta na forma de um gráfico a curva de permanência de um conjunto de dados de vazões.

Os dados escolhidos para servirem a essa aplicação referem-se às vazões diárias de um posto fluviométrico, identificado pelo código 86410000, localizado na bacia do Rio Taquari-Antas, no estado do Rio Grande do Sul, sendo armazenados em um arquivo padrão Excel 4.0. A Figura 4.13 Apresenta o hidrograma do Posto 86410000, com as vazões diárias do período de 01/03/1988 a 31/12/1995.

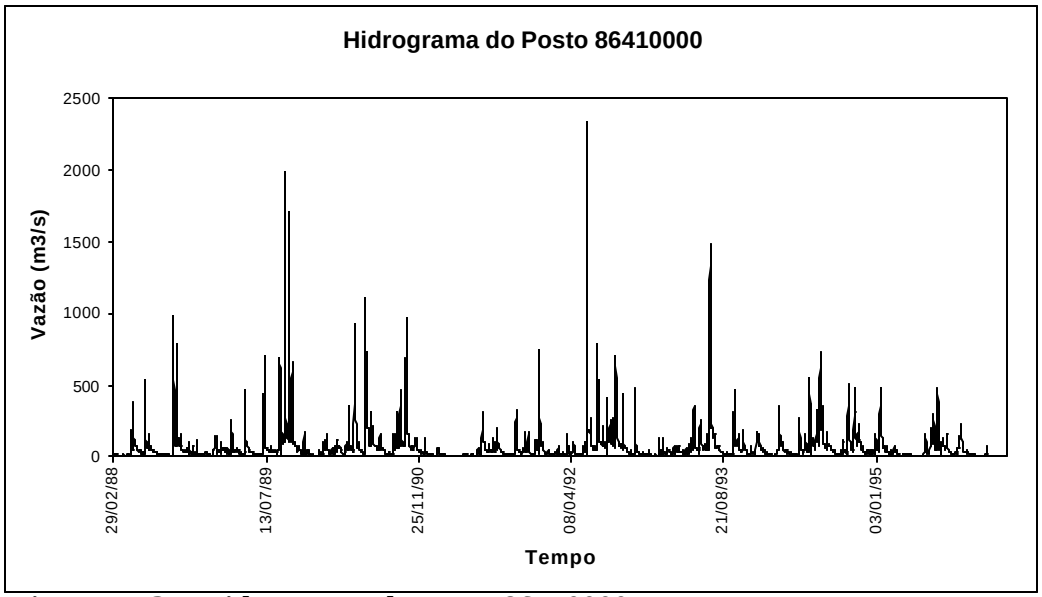


Figura 4.13 – Hidrograma do Posto 86410000.

A Figura 4.14 apresenta a curva de permanência de vazões, resultado da aplicação dessa rotina, utilizando escala logarítmica no eixo das ordenadas.

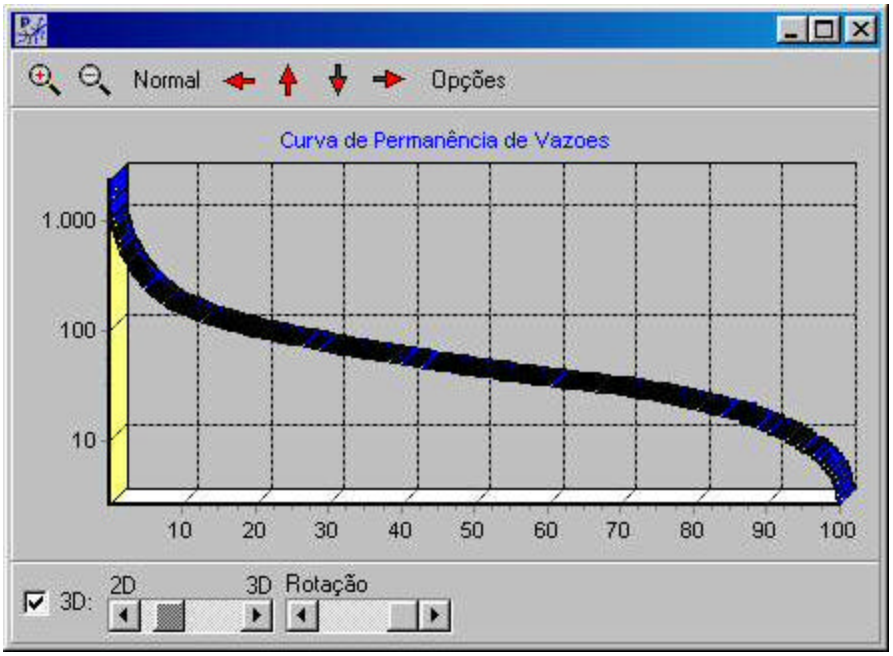


Figura 4.14 – Curva de Permanência de Vazões.

A rotina implementada também gera uma tabela com os dados que produziram a curva apresentada acima, de forma que estes podem ser transferidos para uma planilha eletrônica, como o Excel, por exemplo.

Da mesma forma como foram utilizados dados desvinculados com o projeto em estudo, ao qual está ligado Script Geral, pode-se acessar dados e informações do próprio projeto e realizar alguma análise sem ativar o processo de simulação da rede hidrográfica.

CAPÍTULO 5 - CONCLUSÕES E RECOMENDAÇÕES.

5.1 CONCLUSÕES.

O presente trabalho teve por objetivos a implementação de ferramentas genéricas de planejamento do uso da água e operação de reservatórios no modelo Propagar MOO, utilizando a linguagem Pascal Script, a implementação de um método padrão para simular e quantificar a geração de energia hidrelétrica em uma rede hidrográfica, dentro do modelo, e, a partir deste, propor funções e procedimentos em Pascal Script, específicos para este fim, permitindo ao usuário a implementação e utilização de métodos alternativos.

Dentre os objetivos, pode-se incluir, também, a implementação de ferramentas de auxílio à análise de resultados da simulação no modelo Propagar, bem como a orientação dos usuários do modelo no processo de desenvolvimento de ferramentas aplicadas, utilizando as chamadas de rotinas em Pascal Script que o modelo possui.

No que se refere ao alcance desses objetivos, foi possível, com a realização do presente trabalho, formular as seguintes conclusões:

- O uso da rotina PLANEJA, dentro do modelo Propagar MOO, permite que sejam criadas ferramentas genéricas, escritas na linguagem Pascal Script, para planejar o uso da água e promover a operação de reservatório;

- A utilização de arquivos padrão Excel torna mais ágil a entrada de dados e parâmetros para as rotinas de planejamento, além de simplificar a realização de testes de diferentes parâmetros nas referidas rotinas;

- A aplicação do processo de avaliação de demandas a jusante do reservatório, através da criação de funções específicas para esse fim, no interior das rotinas de planejamento, permite uma definição mais precisa do volume de água a ser liberado do reservatório em cada intervalo de simulação, de forma que, dependendo da disponibilidade hídrica, sejam minimizadas as falhas no atendimento dessas demandas;

- A consideração da precipitação e da evaporação sobre o reservatório influi de maneira significativa na obtenção de resultados precisos durante o processo de balanço hídrico do mesmo, ao ser estimada a sua disponibilidade hídrica. A sua não consideração pode provocar, dependendo do caso, erros grosseiros;

- A implementação de um método padrão, de fácil utilização, para simular a presença de usinas hidrelétrica em pontos de uma rede hidrográfica, com suas novas janelas de entrada de dados, bem como as novas propriedades desses PCs, permitiram quantificar a geração de energia hidrelétrica em qualquer PC, com ou sem reservatório, dentro de uma rede hidrográfica;

- A facilidade de alteração dos parâmetros ligados à geração de energia torna muito simples o processo de testar a implantação de usinas com diferentes características;

- A implementação de funções e procedimentos em Pascal Script, desenvolvidos para serem utilizados na simulação da geração de energia hidrelétrica, permite aos usuários do modelo Propagar MOO a criação e utilização de métodos alternativos para simular e avaliar a energia gerada em um PC da rede;

- O uso da rotina de planejamento que avalia as demandas de energia indicadas pelo usuário, para, posteriormente, definir a liberação de água do reservatório, permite simular a geração de energia como uso prioritário da água desse reservatório e avaliar os efeitos disso nas demais demandas;

- A utilização da Rotina de Uso Geral, executada após a simulação, torna o modelo Propagar MOO bastante versátil para a implementação de ferramentas de auxílio à análise dos resultados de uma simulação;

- Os recursos para a apresentação dos resultados de uma simulação, sejam na forma de textos, em formato de planilhas ou na forma de gráficos, nativos do modelo ou disponibilizados através de comandos em Pascal Script, ampliam as possibilidades do usuário, no momento de avaliar os resultados de um estudo;

- A utilização da rotina “Script Geral”, executada de forma independente da simulação da rede hidrográfica construída dentro do modelo, permite a realizar análises de dados internos ou externos à rede hidrográfica, permitindo, inclusive, o controle da simulação;

- A execução de exemplos de rotinas em Pascal Script, através de rotinas Planeja, Cálculo de Energia, Rotinas de Uso Geral e “Scripts Gerais”, permitiram demonstrar a utilidade e a versatilidade introduzida no modelo Propagar MOO por esse tipo de ferramenta, bem como orientar os usuários do modelo no processo de desenvolvimento de novas ferramentas aplicadas, utilizando essas chamadas de rotinas que o modelo possui;

- A utilização da linguagem Pascal Script para a construção e alteração de ferramentas aplicadas ao modelo é de extrema facilidade para o usuário;

- A linguagem Pascal Script pode ter sua capacidade facilmente ampliada, através da criação e disponibilização de novas funções e métodos.

Em função disso, pode-se concluir que a utilização de ferramentas implementadas através de rotinas Pascal Script amplia, de modo significativo, a capacidade e a flexibilidade do modelo Propagar MOO.

5.2 RECOMENDAÇÕES.

A seguir são apresentadas algumas recomendações que poderão servir de orientação para outros trabalhos que, eventualmente, venham dar continuidade a este estudo.

- Visando aprimorar o processo de avaliação da disponibilidade hídrica dos reservatórios, em rotinas de planejamento implementadas em Pascal Script, recomenda-se utilizar estruturas de cálculo iterativo que considerem a precipitação e a evaporação sobre a superfície dos reservatórios, de forma similar à existente no balanço hídrico de reservatório, dentro da rotina principal do modelo;

- Para melhor avaliar as rotinas de planejamento do uso da água e de operação de reservatórios que possuem algum esquema de racionamento preventivo, recomenda-se a utilização de uma bacia hidrográfica com disponibilidade hídrica natural reduzida, bem como o uso de cenários que apresentem períodos longos de recessão hídrica;

- Com o objetivo de avaliar os resultados obtidos pela simulação da geração de energia em qualquer PC de uma rede hidrográfica, recomenda-se a simulação da presença de usinas hidrelétricas em PCs sem controle de reservatório, o que permitiria, também, uma melhor avaliação da ferramenta que implementa a construção da curva de permanência da energia gerada;

- Para testar o potencial da rotina “Script Geral”, recomenda-se a construção de uma rotina em Pascal Script que realize um processo de otimização por enumeração, através do controle da simulação da rede hidrográfica.

CAPÍTULO 6 - REFERÊNCIAS BIBLIOGRÁFICAS.

BARTH F. T. 1987. *Fundamentos Para Gestão de Recursos Hídricos.* In: Modelos para Gerenciamento de Recursos Hídricos. São Paulo: ABRH: Nobel. p.1-91.

BRAGA Jr., B. P. F. 1987. *Técnicas de Otimização e Simulação Aplicadas em Recursos Hídricos.* In: BARTH et al. Modelos para Gerenciamento de Recursos Hídricos. São Paulo: ABRH: Nobel. p.425-518.

BRAGA Jr., B. P. F., GOBETTI, L. 1997. *Análise Multiobjetivo.* In: PORTO, R. L. et al. Técnicas Quantitativas para o Gerenciamento de Recursos Hídricos. São Paulo: ABRH. p.361-420.

BRASIL. 1997. *Lei 9.433, de 8 de janeiro de 1997: institui a Política Nacional de Recursos Hídricos, cria o Sistema Nacional de Gerenciamento de Recursos Hídricos.* Brasília.

CONCEIÇÃO, A. R. (2000). *Pascal Script - Interpretador e Linguagem*. (contribuição pessoal - não publicado).

ELETROBRÁS. 1985. *Manual de Microcentrais Hidrelétricas.* 530 p.

GOULTER, I. C. 1992. *Systems Analysis in Water - Distribution Network Design: From Theory to Practice.* Journal of Water Resources Planning and Management. New York, v.118, n.3, p.238-248, May/June.

GRULL, D. 1981. *Operação Hidráulica de Reservatórios: Enfoque Probabilístico e Determinístico de Condições de Contorno.* Revista de Hidrologia e Recursos Hídricos. São Paulo, v.3, n.1, p.31-41, Jan/Jun.

HALL, W. A., DRACUP, J. A. 1974. *Ingenieria de Sistemas en los Recursos Hidraulicos.* México: Continental. 436 p.

INSTITUTO DE PESQUISAS ESPACIAIS. 1972. *Engenharia de Sistemas: Planejamento e Controle de Projetos.* 3a.ed. Petrópolis.

LABADIE, J. W. 1987. *Otimização da Operação de Projetos Hidroagrícolas.* [Brasília]: PRONI. 249p.

LANNA, A. E. 1982. *Operação estratégica de reservatórios para suprimento hídrico e controle de cheias.* Porto Alegre: Instituto de Pesquisas Hidráulicas da UFRGS. 123p. (Recursos Hídricos. Publicação n. 4).

LANNA, A. E. 1986. *Dimensionamento e/ou Expansão da Capacidade de um Sistema de Suprimento Hídrico com Técnicas de Programação Linear.* In: Congresso Latino-Americano sobre Métodos Computacionais para Engenharia, 7., 1986, São Carlos. v.2, p.795-808.

LANNA, A. E. 1993. *Gerenciamento de Bacia Hidrográfica: Conceitos, Princípios e Aplicações no Brasil.* Porto Alegre: Instituto de Pesquisas Hidráulicas da UFRGS. 70p. (Recursos Hídricos. Publicação n.29).

LANNA, A. E. 1997a. *Análise Sistêmica dos Recursos Hídricos - HIDP-64.* Porto Alegre: Instituto de Pesquisas Hidráulicas da UFRGS.

LANNA, A. E. 1997b. *Regularização em Reservatórios.* In: TUCCI, C. E. (org.) Hidrologia: Ciência e Aplicação. 2a ed. Porto Alegre: Ed. da Universidade: ABRH. cap.18, p.703-725.

LANNA, A. E. 1997c. *Gestão dos Recursos Hídricos.* In: TUCCI, C. E. (org.) Hidrologia: Ciência e Aplicação. 2a ed. Porto Alegre: Ed. da Universidade: ABRH. cap.19, p.727-768.

LANNA, A. E. 1997d. *SAGBAH – Sistema de Apoio ao Gerenciamento de Bacias Hidrográficas – versão 97*. IPH-UFRGS. 25p.

LANNA, A. E. (1999). *Gestão das Águas*. (texto da disciplina HIDP-78). IPH-UFRGS. Porto Alegre. 235p. http://orion.ufrgs.br/iph/iph.html.

LANNA, A. E. 2000. *Sistemas de Gestão de Recursos Hídricos – Análise de alguns arranjos institucionais.* Revista Ciência & Ambiente. Julho/Dezembro de 2000. Santa Maria. UFSM. p.21-56.

LOUCKS, D. P. 1992. *Water Resources Systems Models: Their Role in Planning.* Journal of Water Resources Planning and Management, New York, v.118, n.3, p.214-223, May./Jun.

LUZ, L. D. da. 1994. *Análise de Critérios Simplificados para Outorga dos Direitos de Uso da Água na Bacia do Rio Grande, Bahia: Uma Análise Multiobjetivo.* Porto Alegre: UFGRS - Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento. 124f. Dissertação (Mestrado).

PORTO, R. L. et al. 1997. *Técnicas Quantitativas para o Gerenciamento de Recursos Hídricos.* São Paulo: ABRH. 420p.

SIMONOVIC, S. P. 1993. *Reservoir Systems Analysis: Closin Gap Between Theory and Practice.* Journal of Water Resources Planning and Management, New York, v.118, n.3, p.262-280, May./June.

VIANNA JR, W. P. 1998. *Operação Ótima da Sistema de Reservatórios da Bacia do Rio Curu - CE*. Dissertação de Mestrado. Curso de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental. IPH-UFRGS. Porto Alegre. 116p.

VIEGAS Fº, J. S., LANNA, A. E. 1999a. *PROPAGAR 2000 - Manual do Usuário 44p.*. Porto Alegre: IPH-UFRGS e FEA/IFM-UFPel. 44p.

VIEGAS Fº, J. S., 1999b. *Modelagem Orientada a Objetos Aplicada a Sistemas de Recursos Hídricos.* Trabalho apresentado como parte dos requisitos para Exame de Qualificação. Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental. IPH-UFRGS. Porto Alegre. 93p.

VIEGAS Fº, J. S., 2000. *O Paradigma da Modelagem Orientada a Objetos Aplicado a Sistemas de Apoio à Decisão em Sistemas de Recursos Hídricos.* Tese de Doutorado. Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Saneamento Ambiental. IPH-UFRGS. Porto Alegre. 294p.

VIEGAS Fº, J. S., LANNA A. E. L., CONCEIÇÃO A. R., 2001. *A Linguagem Pascal Script e sua Aplicação ao Propagar MOO.* In: XIII Simpósio Brasileiro de Recursos Hídricos. Aracaju.

WURBS, R. A. 1993. *Reservoir-System Simulation and Optimization Models.* Journal of Water Resources Planning and Management, New York, v.119, n.4, p.455-472, July/Aug.

YEH, W. W-G. 1985. *Reservoir Management and Operations Models: A State-of-the-art Review.* Water Resources Research, Washington, v.21, n.12, p.1797-1818, Dec.

ANEXO 1 - ROTINAS IMPLEMENTADAS EM PASCAL SCRIPT

Anexo 1.1 ROTINA DE PLANEJAMENTO PARA APLICAÇÃO DA REGRA DE DEFLUÊNCIAS FIXAS COMBINADA COM A REGRA PADRÃO

```
program PlanejaDefluFixa;

var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
    // Declaração das variáveis auxiliares
    PCReserv     : string;  // Var. p/ o nome do PC reservatório
    vDefluFixo   : object;  // Vetor p/ os 12 valores de defluências fixas
    DefluFixo    : real;    // Var. p/ a defluências fixa no cálculo
    DefluFixoHm3 : real;    // Var. p/ a defluências fixas em Hm3/dt
    vVEspera     : object;  // Vetor p/ os 12 valores de Volume de Espera
    VEspera      : real;    // Var. p/ conter o Vol. de Espera no cálculo
    Mes          : integer; // Var. p/ conter o mês referente ao dt atual
    Ano          : integer; // Var. p/ conter o ano referente ao dt atual
    PCR          : object;  // Var. p/ conter o objeto PCR (reservatório)
    PCMontante   : object;  // Objeto PC a montante
    Plan         : object;  // Planilha p/ conter dados do usuário
    dt           : integer; // Intervalo de Tempo
    II           : integer; // Var. Auxiliar p/ índice
    AfluHm3          : real; // Afluências ao resevatório
    VazaoPCsMontante : real; // Vazões dos PCs a montante
    VolInicInterv    : real; // Volume do reservatório no início do dt
    VolDisp      : real; // Volume disponível no reserv. no início do dt
    CapUtil      : real; // Capacidade Util do Reservatório
    Disp         : real; // Disponibilidade hídrica no intervalo de tempo
    L            : real; // Vazão de água a liberar do reservatório no dt
    Vertimento : real; // Variável. p/ conter o Vertimento do Reservatório
    Int_Simul        : integer; // Número de intervalos de simulação

begin
  dt := Projeto.ObtemDeltaT; // Obtém Delta-T atual e atribui à var. dt
  Int_Simul := Projeto.Total_IntSim; // Número de intervalos de simulação
  if dt = 1 then
    begin
  // Dados do usuário - Via arq.EXCELL (Nomes dos PCs e parâmetros)
    Plan := CreateObject(TPlanilha);
    Plan.LoadFromFile(' D:\Diretorio\Arquivo.xls');
    PCReserv   := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
    vDefluFixo := Plan.RowToVec(3,1,12); // Carrega os 12 deflúvios fixos
    vVEspera   := Plan.RowToVec(5,1,12); // Carrega 12 Vol. de Espera
    GlobalObjects.Add('vDefluFixo', vDefluFixo); // Soma à lista global
    GlobalObjects.Add('vVEspera', vVEspera); // Adiciona-o à lista global
    FreeObject(Plan);
    end
  else
    begin // Recupera variáveis globais
    vDefluFixo := TwsDFVec(GlobalObjects.Get('vDefluFixo'));
    vVEspera   := TwsDFVec(GlobalObjects.Get('vVEspera'));
    end;
  PCR := Projeto.PCPeloNome(PCReserv); // Atribui o Reserv. à var. PCR
  if dt = 1 then VolInicInterv := PCR.VolumeInicial
            else VolInicInterv := PCR.ObtemVolume(dt-1);
// Cálculo da disponibilidade hídrica
  VolDisp := VolInicInterv - PCR.VolumeMinimo;
  AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
  VazaoPCsMontante := 0;
  if PCR.PCs_aMontante > 0 then
```

```
     begin
     for II := 0 to (PCR.PCs_aMontante - 1) do
         begin
         PCMontante := PCR.PC_aMontante(II);
         Projeto.RealizaBalancoHidricoAte(PCMontante);
         VazaoPCsMontante := VazaoPCsMontante +
                             PCMontante.ObtemDefluencia(dt);
         end;
     VazaoPCsMontante := PCR.m3_Hm3_Intervalo(VazaoPCsMontante);
     end;
  Disp := VolDisp + AfluHm3 + VazaoPCsMontante;
// Cálculo da capacidade útil do reservatório
  CapUtil := PCR.VolumeMaximo-PCR.VolumeMinimo;
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
// Extrai o volume de espera (mês atual) do vetor fornecido pelo usuário
  VEspera := (vVEspera.Get(Mes) * PCR.VolumeMaximo / 100);
// Extrai a defluência fixa (mês atual) do vetor fornecido pelo usuário
  DefluFixo := vDefluFixo.Get(Mes);
  DefluFixoHm3 := PCR.m3_Hm3_Intervalo(DefluFixo); // Transforma em Hm3/dt
  if Disp <= 0 then L := 0 // Verifica a disp. hídrica p/ atender demandas
  else
     if Disp <= (CapUtil - VEspera) then L := DefluFixo
     else
        begin
        Vertimento := Disp - (CapUtil - VEspera);
        if Vertimento < DefluFixoHm3 then L := DefluFixo
        else L := PCR.Hm3_m3_Intervalo(Vertimento); // Converte p/ m3
        end;
{O valor a ser liberado é atribuído à propriedade (variável)
DefluvioPlanejado no intervalo dt, ou seja, DeltaT atual}
  PCR.AtribuiDefluvioPlanejado(dt, L);
// Escrita de um Arquivo LOG
     Saida.WriteLN('Planeja');
     Saida.WriteLN('Intervalo    : ' + ValToStr(dt));
     Saida.WriteLN('PCReserv     : ' + PCReserv);
     Saida.WriteLN('DefluFixo    : ' + ValToStr(DefluFixo));
     Saida.WriteLN('VEspera      : ' + ValToStr(VEspera));
     Saida.WriteLN('Vol.Inic. dt : ' + ValToStr(VolInicInterv));
     Saida.WriteLN('Vol. Disp.   : ' + ValToStr(VolDisp));
     Saida.WriteLN('Vol. Max.    : ' + ValToStr(PCR.VolumeMaximo));
     Saida.WriteLN('Vol. Min.    : ' + ValToStr(PCR.VolumeMinimo));
     Saida.WriteLN('CapUtil      : ' + ValToStr(CapUtil));
     Saida.WriteLN('Disp         : ' + ValToStr(Disp));
     Saida.WriteLN('Vertimento   : ' + ValToStr(Vertimento));
     Saida.WriteLN('L            : ' + ValToStr(L));
  if dt = Projeto.Total_IntSim then
    begin
    GlobalObjects.Remove('vDefluFixo');
    GlobalObjects.Remove('vVEspera');
    Saida.Show;
    end;
end.
```

Anexo 1.2 ROTINA DE PLANEJAMENTO PARA A APLICAÇÃO DA REGRA PADRÃO MODIFICADA COM AVALIAÇÃO DE DEMANDAS

```
program PlanejaRegraPadraoModificada;

var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
// Declaração das variáveis auxiliares
   PCReserv   : string;  // Var. p/ conter o nome do primeiro PC da lista
   PCDemandas : string;  // Var. p/ conter o nome do último PC da lista
   vListaPCs  : object;  // Vetor p/ conter PCs do Reserv. até as demandas
   N          : integer; // Número de PCs da lista
   I          : integer; // Var. p/ indexar a lista
   II         : integer; // Var. p/ indexar
   vAlfa      : object; // Vetor p/ conter os 12 valores Alfa da regra
   vBeta      : object;  // Vetor p/ conter os 12 Parâmetros Beta da regra
   Alfa       : real;    // Var. p/ conter o Parâmetro Alfa p/ o cálculo
   Beta       : real;    // Var. p/ conter o Parâmetro Beta p/ o cálculo
   Mes        : integer; // Var. p/ conter o mês referente ao dt atual
   Ano        : integer; // Var. p/ conter o ano referente ao dt atual
   PC         : object;  // Var. p/ conter o objeto PC (qualquer PC)
   PCAnt      : object;  // Var. p/ conter o objeto PC Anterior
   PCR        : object;  // Var. p/ conter o objeto PCR (reservatório)
   PCMontante : object;  // Var. p/ conter o objeto PC (qualquer PC)
   Plan       : object;  // Planilha p/ conter dados de entrada do usuário
   dt         : integer; // Intervalo de Tempo
   DemReservatorio  : real; // Demandas do PC Reservatório
   Dem1, Dem2, Dem3 : real; // Demandas prim., sec. e terciária de cada PC
   DemAnt1, DemAnt2, DemAnt3 : real; // Demandas do PC Anterior (PCAnt)
   RetAnt1, RetAnt2, RetAnt3 : real; // Fração de retorno dessas demandas
   SB         : real; // Afluência das sub-bacias aos PCs
   Aux  : real; // Var. Auxiliar para acumular demandas remanescentes
   DemResidual : real; // Demanda não atendida pelas afluências das bacias
   DemResidualHm3 : real; // Demanda não atendida pelas afluências em Hm3
   AfluHm3 : real; // Afluência total ao reservatório em Hm3
   VazaoPCsMontante : real; // Var. p/ conter vazoes dos PCs à montante
   VolInicInterv    : real; // Volume do reservatório no início do dt
   VolDisp : real; // Volume disponível no reservatório no início do dt
   CapUtil : real; // Capacidade Util do Reservatório (VOLmax-VOLmin)
   Disp : real; // Disponibilidade hídrica no intervalo de tempo
   L    : real; // Vazão de água a liberar do reservatório no dt
   Vertimento : real; // Vertimento no reservatório no dt
   Int_Simul  : integer; // Número de intervalos de simulação

begin
  dt := Projeto.ObtemDeltaT; // Obtem Delta-T atual e atribui à var. dt
  Int_Simul := Projeto.Total_IntSim; // Número de intervalos de simulação
  if dt = 1 then
     begin
   // Dados do usuário - Via arquivo EXCELL
     Plan := CreateObject(TPlanilha);
     Plan.LoadFromFile(' D:\Diretorio\Arquivo.xls');
     PCReserv := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
     PCDemandas := Plan.GetEntry(1,2); // Ret. string da L1 e C2 da Plan.
     vAlfa := Plan.RowToVec(2,1,12); // Cria vetor com 12 parâmetros Alfa
     vBeta := Plan.RowToVec(3,1,12); // Cria vetor com 12 parâmetros Beta
     // Criação da lista de PCs apartir dos pontos dados pelo usuário
     vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);
     // Adiciona à lista de variáveis globais
     GlobalObjects.Add('vListaPCs', vListaPCs);
```

```
     GlobalObjects.Add('vAlfa', vALfa);
     GlobalObjects.Add('vBeta', vBeta);
     FreeObject(Plan);
     end
  else
     begin // Recupera variáveis globais
     vListaPCs  := TStringList(GlobalObjects.Get('vListaPCs'));
     vAlfa      := TwsDFVec(GlobalObjects.Get('vAlfa'));
     vBeta      := TwsDFVec(GlobalObjects.Get('vBeta'));
     end;
  PCR := TprPCP(vListaPCs.GetObject(0)); // Atribui o Reserv. à var. PCR
  if dt = 1 then VolInicInterv := PCR.VolumeInicial
            else VolInicInterv := PCR.ObtemVolume(dt-1);
// Cálculo da disponibilidade hídrica
  VolDisp := VolInicInterv - PCR.VolumeMinimo;
  AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
  VazaoPCsMontante := 0;
  if PCR.PCs_aMontante > 0 then
     begin
     for II := 0 to (PCR.PCs_aMontante - 1) do
         begin
         PCMontante := PCR.PC_aMontante(II);
         Projeto.RealizaBalancoHidricoAte(PCMontante);
         VazaoPCsMontante := VazaoPCsMontante +
                             PCMontante.ObtemDefluencia(dt);
         end;
     VazaoPCsMontante := PCR.m3_Hm3_Intervalo(VazaoPCsMontante);
     end;
  Disp := VolDisp + AfluHm3 + VazaoPCsMontante;
// Cálculo da capacidade do reservatório
  CapUtil := PCR.VolumeMaximo - PCR.VolumeMinimo;
// Cálculo da demanda do Reservatório
 DemReservatorio := 0;
 For II := 1 to 3 do
  DemReservatorio := DemReservatorio + PCR.ObtemValorDemanda(dt, II, 'T');
 DemReservatorio := PCR.m3_Hm3_Intervalo(DemReservatorio);
// Cálculo das demandas residuais
  N := vListaPCs.Count; // Obtem o número de elementos da lista de PCs
  Aux := 0; // Primeiro valor da variável auxiliar
  for I := 1 to (N-1) do
    begin
    PC := TprPCP(vListaPCs.GetObject(I)); // Atrib.PC índice I à var. PC
    Dem1 := PC.ObtemValorDemanda(dt, 1, 'T'); // Demanda Prim. Total no dt
    Dem2 := PC.ObtemValorDemanda(dt, 2, 'T'); // Demanda Sec. Total no dt
    Dem3 := PC.ObtemValorDemanda(dt, 3, 'T'); // Demanda Terc.Total no dt
    SB := PC.ObtemVazaoAfluenteSBs; // Vazão Afluente das Sub-bacias no dt
  // Atribui o PC de índice (I-1) à variável PCAnt
    PCAnt := TprPCP(vListaPCs.GetObject(I-1));
    DemAnt1 := PCAnt.ObtemValorDemanda(dt, 1, 'T');
    DemAnt2 := PCAnt.ObtemValorDemanda(dt, 2, 'T');
    DemAnt3 := PCAnt.ObtemValorDemanda(dt, 3, 'T');
  // Fator de retorno das demandas do PC anterior
    RetAnt1 := PCAnt.FatorDeRetorno(1);
    RetAnt2 := PCAnt.FatorDeRetorno(2);
    RetAnt3 := PCAnt.FatorDeRetorno(3);
    VazaoPCsMontante := 0;
    if PC.PCs_aMontante > 1 then
       begin
         for II := 0 to (PC.PCs_aMontante - 1) do
           begin
           PCMontante := PC.PC_aMontante(II);
           if PCMontante <> PCAnt then
```

```
              begin
              Projeto.RealizaBalancoHidricoAte(PCMontante);
              VazaoPCsMontante := VazaoPCsMontante +
                                  PCMontante.ObtemDefluencia(dt);
              end;
            end;
        end;
  // Cálculo da demanda não atendida pelas afluências das sub-bacias
    DemResidual := DemResidual + (Dem1 + Dem2 + Dem3) - SB –
                   VazaoPCsMontante - (DemAnt1 * RetAnt1 + DemAnt2 *
                   RetAnt2 + DemAnt3 * RetAnt3);
    Aux := Max(Aux,Max(DemResidual,0));
    end;
  DemResidual := Aux;
  DemResidualHm3 := PC.m3_Hm3_Intervalo(DemResidual);
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
  // Alfa recebe o % do Vol Max do reserv., na posição "Mes" de vAlfa
  Alfa := PCR.VolumeMaximo*(vAlfa.Get(Mes))/100;
  // Beta recebe o % do Vol Max do reserv., na posição "Mes" de vBeta
  Beta := PCR.VolumeMaximo*(vBeta.Get(Mes))/100;
{Quando a água geranda nas próprias sub-bacias atenderem as demandas o
valor de DemResidual será negativo. Então faz-se L = 0}
  if DemResidual <= 0 then
    begin
    if (Disp – DemReservatorio) <= (CapUtil - Beta) then L := 0
    else L := PCR.Hm3_m3_Intervalo((Disp – DemReservatorio) –
              (CapUtil - Beta));
    end
  else
    if (Disp – DemReservatorio) <= PCR.VolumeMinimo then L := 0
    else
      if (Disp – DemReservatorio) < (DemResidualHm3 + Alfa) then
        L := PCR.Hm3_m3_Intervalo((Disp – DemReservatorio) *
             DemResidualHm3 / (DemResidualHm3 + Alfa))
      else
        if (Disp – DemReservatorio) <= (CapUtil - Beta) then
            L := DemResidual
        else
          begin
          Vertimento := (Disp – DemReservatorio) - (CapUtil - Beta);
          if Vertimento < DemResidualHm3 then L := DemResidual
          else L := PCR.Hm3_m3_Intervalo(Vertimento);
          end;
  PCR.AtribuiDefluvioPlanejado(dt, L);
// Escrita do Arquivo LOG
  Saida.WriteLN('Intervalo  : ' + ValToStr(dt));
  Saida.WriteLN('Alfa       : ' + ValToStr(Alfa));
  Saida.WriteLN('Beta       : ' + ValToStr(Beta));
  Saida.WriteLN('Vol. Disp. : ' + ValToStr(VolDisp));
  Saida.WriteLN('Vz. Aflu.  : ' + ValToStr(PCR.ObtemVazaoAfluenteSBs));
  Saida.WriteLN('Disp       : ' + ValToStr(Disp));
  Saida.WriteLN('DemResiualdHm3 : ' + ValToStr(DemResidualHm3));
  Saida.WriteLN('Vertimento     : ' + ValToStr(Vertimento));
  Saida.WriteLN('L              : ' + ValToStr(L));
  if dt = Projeto.Total_IntSim then
    begin
    GlobalObjects.Remove('vListaPCs');
    GlobalObjects.Remove('vAlfa');
    GlobalObjects.Remove('vBeta');
    Saida.Show;
    end;
end.
```

Anexo 1.3 ROTINA DE PLANEJAMENTO PARA A APLICAÇÃO DA REGRA DOS VOLUMES-METAS COM AVALIAÇÃO DE DEMANDAS

```
program PlanejaVolumes-Metas;

var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
// Declaração das variáveis auxiliares
    PCReserv   : string;  // Var. p/ conter o nome do primeiro PC da lista
    PCDemandas : string;  // Var. p/ conter o nome do último PC da lista
    vListaPCs  : object;  // Vetor p/ conter PCs do Reserv. até demandas
    N          : integer; // Número de PCs da lista
    I          : integer; // Var. p/ indexar a lista
    II         : integer; // Var. p/ indexar
    vVolMetMax : object;  // Vetor p/ conter os 12 valores de volume máx.
    vVolMetMin : object;  // Vetor p/ conter os 12 valores de volume mín.
    VolMetMax  : real;    // Var. p/ conter o Vol. máximo p/ o cálculo
    VolMetMin  : real;    // Var. p/ conter o Vol. máximo p/ o cálculo
    Mes        : integer; // Var. p/ conter o mês referente ao dt atual
    Ano        : integer; // Var. p/ conter o ano referente ao dt atual
    PC         : object;  // Var. p/ conter o objeto PC (qualquer PC)
    PCAnt : object;  // Var. p/ conter o objeto PC Anterior a qualquer PC
    PCR   : object;  // Var. p/ conter o objeto PCR (reservatório)
    PCMontante : object;  // Var. p/ conter o PC à montante
    Plan  : object; // Var. Planilha p/ conter dados de entrada do usuário
    dt    : integer; // Intervalo de Tempo
    DemReservatorio  : real; // Demandas do PC Reservatório
    Dem1, Dem2, Dem3 : real; // Demandas prim., sec. e terc. de cada PC
    DemAnt1, DemAnt2, DemAnt3 : real; // Demandas do PC Anterior (PCAnt)
    RetAnt1, RetAnt2, RetAnt3 : real; // Fração de retorno dessas demandas
    SB               : real; // Afluência das sub-bacias aos PCs
    Aux              : real; // Var. Auxiliar p/ conter demandas residuais
    DemResidual      : real; // Demanda não atendida pelas bacias
    DemResidualHm3   : real; // Demanda não atendida em Hm3
    AfluHm3          : real; // Afluência total ao reservatório em Hm3
    VazaoPCsMontante : real; // vazões de PCs à montante
    VolInicInterv    : real; // Volume do reservatório no início do dt
    VolDisp          : real; // Volume disp. no reserv. no início do dt
    Disp  : real; // Disponibilidade hídrica no intervalo de tempo
    L     : real; // Vazão de água a liberar do reservatório no dt
    Vertimento       : real; // Vertimento no reservatório no dt
    Int_Simul        : integer; // Número de intervalos de simulação
begin
  dt := Projeto.ObtemDeltaT; // Obtem Delta-T atual e atribui à var. dt
  Int_Simul := Projeto.Total_IntSim; // Número de intervalos de simulação
  if dt = 1 then
     begin
     // Entrada de dados do usuário, via arq.EXCELL
     Plan := CreateObject(TPlanilha);
     Plan.LoadFromFile(' D:\Diretorio\Arquivo.xls ');
     PCReserv := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
     PCDemandas := Plan.GetEntry(1,2); // Ret. string da L1 e C2 da Plan.
     vVolMetMax := Plan.RowToVec(3,1,12); // Vetor com valores de Vol.Max.
     vVolMetMin := Plan.RowToVec(4,1,12); // Vetor com valores de Vol.Min.
  // Criação da lista de PCs apartir dos pontos dados pelo usuário
     vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);
  // Adiciona vetores na lista de variáveis globais
     GlobalObjects.Add('vListaPCs', vListaPCs);
     GlobalObjects.Add('vVolMetMax', vVolMetMax);
     GlobalObjects.Add('vVolMetMin', vVolMetMin);
     FreeObject(Plan);
```

```
      end
   else
      begin // Recupera variáveis globais
      vListaPCs  := TStringList(GlobalObjects.Get('vListaPCs'));
      vVolMetMax := TwsDFVec(GlobalObjects.Get('vVolMetMax'));
      vVolMetMin := TwsDFVec(GlobalObjects.Get('vVolMetMin'));
      end;
   PCR := TprPCP(vListaPCs.GetObject(0)); // Atribui o Reserv. à var. PCR
   if dt = 1 then VolInicInterv := PCR.VolumeInicial
             else VolInicInterv := PCR.ObtemVolume(dt-1);
   // Cálculo da disponibilidade hídrica
   VolDisp := VolInicInterv - PCR.VolumeMinimo;
   AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
   VazaoPCsMontante := 0;
   if PCR.PCs_aMontante > 0 then
      begin
      for II := 0 to (PCR.PCs_aMontante - 1) do
          begin
          PCMontante := PCR.PC_aMontante(II);
          Projeto.RealizaBalancoHidricoAte(PCMontante);
          VazaoPCsMontante := VazaoPCsMontante +
                              PCMontante.ObtemDefluencia(dt);
          end;
      VazaoPCsMontante := PCR.m3_Hm3_Intervalo(VazaoPCsMontante);
      end;
   Disp := VolDisp + AfluHm3 + VazaoPCsMontante;
// Cálculo da demanda do Reservatório
 DemReservatorio := 0;
 For II := 1 to 3 do
  DemReservatorio := DemReservatorio + PCR.ObtemValorDemanda(dt, II, 'T');
 DemReservatorio := PCR.m3_Hm3_Intervalo(DemReservatorio);
// Cálculo das demandas residuais
   N := vListaPCs.Count; // Obtem o número de elementos da lista de PCs
   Aux := 0; // Atribui primeiro valor à variável auxiliar
   for I := 1 to (N-1) do
     begin
     PC := TprPCP(vListaPCs.GetObject(I)); // PC de índice I p/ var. PC
     Dem1 := PC.ObtemValorDemanda(dt, 1, 'T'); // Demanda Prim. Total no dt
     Dem2 := PC.ObtemValorDemanda(dt, 2, 'T'); // Demanda Sec. Total no dt
     Dem3 := PC.ObtemValorDemanda(dt, 3, 'T'); // Demanda Terc. Total no dt
     SB := PC.ObtemVazaoAfluenteSBs; // Vazão Afluente das Sub-bacias no dt
     PCAnt := TprPCP(vListaPCs.GetObject(I-1)); // PC de índice I-1
     DemAnt1 := PCAnt.ObtemValorDemanda(dt, 1, 'T');
     DemAnt2 := PCAnt.ObtemValorDemanda(dt, 2, 'T');
     DemAnt3 := PCAnt.ObtemValorDemanda(dt, 3, 'T');
     RetAnt1 := PCAnt.FatorDeRetorno(1); // Fator de Retorno do PCAnt
     RetAnt2 := PCAnt.FatorDeRetorno(2); // Fator de Retorno do PCAnt
     RetAnt3 := PCAnt.FatorDeRetorno(3); // Fator de Retorno do PCAnt
     VazaoPCsMontante := 0;
     if PC.PCs_aMontante > 1 then
        begin
          for II := 0 to (PC.PCs_aMontante - 1) do
            begin
            PCMontante := PC.PC_aMontante(II);
            if PCMontante <> PCAnt then
               begin
               Projeto.RealizaBalancoHidricoAte(PCMontante);
               VazaoPCsMontante := VazaoPCsMontante +
                                   PCMontante.ObtemDefluencia(dt);
               end;
            end;
        end;
   // Cálculo da demanda não atendida pelas afluências das sub-bacias
```

```
    DemResidual := DemResidual + (Dem1 + Dem2 + Dem3) - SB –
                   VazaoPCsMontante - (DemAnt1 * RetAnt1 + DemAnt2 *
                   RetAnt2 + DemAnt3 * RetAnt3);
    Aux := Max(Aux,Max(DemResidual,0));
    end;
  DemResidual := Aux;
  DemResidualHm3 := PC.m3_Hm3_Intervalo(DemResidual);
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
  VolMetMax := PCR.VolumeMaximo*vVolMetMax.Get(Mes)/100 –
               PCR.VolumeMinimo; // Calc. o VolMetMax atual
  VolMetMin := PCR.VolumeMaximo*vVolMetMin.Get(Mes)/100 –
               PCR.VolumeMinimo; // Calc. o VolMetMin atual
  if VolMetMin < 0 then VolMetMin := 0;
{Quando a água geranda nas próprias sub-bacias atenderem as demandas o
valor de DemResidual será negativo. Então faz-se L = 0}
  if DemResidual <= 0 then
     begin
     if (Disp – DemReservatorio) <= VolMetMax then L := 0
     else
       begin
       Vertimento := ((Disp – DemReservatorio) - VolMetMax);
       L := PCR.Hm3_m3_Intervalo(Vertimento);
       end;
     end
  else
    if (Disp – DemReservatorio) < VolMetMin then L := 0
    // Verifica a disp. hídrica para atender a regra de VolMetas Mínimos
    else
      if ((Disp – DemReservatorio) - DemResidualHm3) < VolMetMin then
         L := PCR.Hm3_m3_Intervalo((Disp – DemReservatorio) - VolMetMin)
      else
        if (Disp – DemReservatorio) <= VolMetMax then L := DemResidual
        else
          begin
          Vertimento := (Disp – DemReservatorio) - VolMetMax;
          if Vertimento < DemResidualHm3 then L := DemResidual
          else L := PCR.Hm3_m3_Intervalo(Vertimento); // Conv. p/ m3
          end;
{O valor a ser liberado é atribuído à propriedade (variável)
DefluvioPlanejado no intervalo dt, ou seja, DeltaT atual}
  PCR.AtribuiDefluvioPlanejado(dt, L);
// Escrita do Arquivo LOG
     Saida.WriteLN('Intervalo         : ' + ValToStr(dt));
     Saida.WriteLN('VolMetMax         : ' + ValToStr(VolMetMax));
     Saida.WriteLN('VolMetMin         : ' + ValToStr(VolMetMin));
     Saida.WriteLN('Disp              : ' + ValToStr(Disp));
     Saida.WriteLN('DemResiualdHm3    : ' + ValToStr(DemResidualHm3));
     Saida.WriteLN('L                 : ' + ValToStr(L));
  if dt = Projeto.Total_IntSim then
    begin
    GlobalObjects.Remove('vListaPCs');
    GlobalObjects.Remove('vVolMetMax');
    GlobalObjects.Remove('vVolMetMin');
    Saida.Show;
    end;
end.
```

Anexo 1.4 Rotina de Planejamento para a Aplicação da Regra dos Volumes-Metas com Avaliação de Demandas, Considerando a Precipitação e a Evaporação Sobre o Reservatório

```
...

program PlanejaVolumesMetas;
var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
// Declaração das variáveis auxiliares
  PCReserv   : string;  // Var. p/ conter o nome do primeiro PC da lista
  PCDemandas : string;  // Var. p/ conter o nome do último PC da lista
  vListaPCs  : object; // Vetor p/ conter os PCs do Reserv. até demandas
  N          : integer; // Número de PCs da lista
  I          : integer; // Var. p/ indexar a lista
  II         : integer; // Var. p/ indexar
  vVolMetMax : object; // Vetor p/ conter os Parâmetros de volume máximo
  vVolMetMin : object; // Vetor p/ conter os Parâmetros de volume mínimo
  VolMetMax  : real;    // Var. p/ conter o Vol. máximo p/ o cálculo
  VolMetMin  : real;    // Var. p/ conter o Vol. máximo p/ o cálculo
  Mes        : integer; // Var. p/ conter o mês referente ao dt atual
  Ano        : integer; // Var. p/ conter o ano referente ao dt atual
  PC         : object;  // Var. p/ conter o objeto PC (qualquer PC)
  PCAnt      : object;  // Var. p/ conter o PC Anterior a qualquer PC
  PCR        : object;  // Var. p/ conter o objeto PCR (reservatório)
  PCMontante : object;  // Var. p/ conter o PC à montante
  Plan       : object;  // Var. Planilha p/ conter dados de entrada
  dt         : integer; // Intervalo de Tempo
  DemReservatorio  : real; // Demandas no reservatório
  Dem1, Dem2, Dem3 : real; // Demandas prim., sec. e terciária de cada PC
  DemAnt1, DemAnt2, DemAnt3 : real; // Demandas do PC Anterior (PCAnt)
  RetAnt1, RetAnt2, RetAnt3 : real; // Fração de retorno dessas demandas
  SB               : real; // Afluência das sub-bacias aos PCs
  Aux              : real; // Var. Auxiliar p/ conter demandas residuais
  DemResidual      : real; // Demanda não atendida pelas bacias
  DemResidualHm3   : real; // Demanda não atendida pelas afluências em Hm3
  AfluHm3          : real; // Afluência total ao reservatório em Hm3
  VazaoPCsMontante : real; // vazões de PCs à montante
  VazaoPCsMontanteRes : real; // vazões de PCs à montante do Reservatório
  VolInicInterv    : real; // Volume do reservatório no início do dt
  L                : real; // Vazão de água a liberar do reservatório
  Vertimento       : real; // Vertimento no reservatório no dt
  Int_Simul        : integer; // Número de intervalos de simulação
  VolAtual : real;
  VolMedio : real;
  VolFinal : real;
  AreaMedia : real;
  AfluAux : real;
  CicloEvapo : integer;
  Difer : real;

begin
  dt := Projeto.ObtemDeltaT; // Obtem Delta-T atual
  Int_Simul := Projeto.Total_IntSim; // Obtem o núm. de interv. simulação
  if dt = 1 then
     begin
// Entrada de dados do usuário, via arq.EXCELL
     Plan := CreateObject(TPlanilha);
     Plan.LoadFromFile('D:\Tese\Simulacao\EntradaXLS\VolumeMetas.xls');
     PCReserv := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
```

```
     PCDemandas := Plan.GetEntry(1,2); // Ret. string da L1 e C2 da Plan.
     vVolMetMax := Plan.RowToVec(3,1,12); // vetor com parâmetros Vol.Max.
     vVolMetMin := Plan.RowToVec(4,1,12); // vetor com parâmetros Vol.Min.
// Criação da lista de PCs apartir dos pontos dados pelo usuário
     vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);
     GlobalObjects.Add('vListaPCs', vListaPCs);
     GlobalObjects.Add('vVolMetMax', vVolMetMax);
     GlobalObjects.Add('vVolMetMin', vVolMetMin);
     FreeObject(Plan);
     end
  else
    begin   // Recupera var. globais
    vListaPCs  := TStringList(GlobalObjects.Get('vListaPCs'));
    vVolMetMax := TwsDFVec(GlobalObjects.Get('vVolMetMax'));
    vVolMetMin := TwsDFVec(GlobalObjects.Get('vVolMetMin'));
    end;
  PCR := TprPCP(vListaPCs.GetObject(0)); // Atribui o Reserv. à var. PCR
  if dt = 1 then VolInicInterv := PCR.VolumeInicial
            else VolInicInterv := PCR.ObtemVolume(dt-1);
  AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
  VazaoPCsMontanteRes := 0;
  if PCR.PCs_aMontante > 0 then
     begin
     for II := 0 to (PCR.PCs_aMontante - 1) do
         begin
         PCMontante := PCR.PC_aMontante(II);
         Projeto.RealizaBalancoHidricoAte(PCMontante);
         VazaoPCsMontanteRes := VazaoPCsMontanteRes +
         PCMontante.ObtemDefluencia(dt);
         end;
     VazaoPCsMontanteRes := PCR.m3_Hm3_Intervalo(VazaoPCsMontanteRes);
     end;
  N := vListaPCs.Count; // Obtem o número de elementos da lista de PCs
  Aux := 0; // Atribui primeiro valor à variável auxiliar
  for I := 1 to (N-1) do
    begin
    PC := TprPCP(vListaPCs.GetObject(I)); // Atrib.o PC índice I à var. PC
    Dem1 := PC.ObtemValorDemanda(dt, 1, 'T');
    Dem2 := PC.ObtemValorDemanda(dt, 2, 'T');
    Dem3 := PC.ObtemValorDemanda(dt, 3, 'T');
    SB   := PC.ObtemVazaoAfluenteSBs; // Vazão Aflu. das Sub-bacias no dt
    PCAnt := TprPCP(vListaPCs.GetObject(I-1));
    DemAnt1 := PCAnt.ObtemValorDemanda(dt, 1, 'T');
    DemAnt2 := PCAnt.ObtemValorDemanda(dt, 2, 'T');
    DemAnt3 := PCAnt.ObtemValorDemanda(dt, 3, 'T');
    RetAnt1 := PCAnt.FatorDeRetorno(1);
    RetAnt2 := PCAnt.FatorDeRetorno(2);
    RetAnt3 := PCAnt.FatorDeRetorno(3);
    VazaoPCsMontante := 0;
    if PC.PCs_aMontante > 1 then
      begin
      for II := 0 to (PC.PCs_aMontante - 1) do
        begin
        PCMontante := PC.PC_aMontante(II);
        if PCMontante <> PCAnt then
          begin
          Projeto.RealizaBalancoHidricoAte(PCMontante);
          VazaoPCsMontante := VazaoPCsMontante +
                              PCMontante.ObtemDefluencia(dt);
          end;
        end;
      end;
// Cálculo da demanda não atendida pelas afluências das sub-bacias
```

```
    DemResidual := DemResidual + (Dem1 + Dem2 + Dem3) - SB –
                   VazaoPCsMontante - (DemAnt1 * RetAnt1 + DemAnt2
                   * RetAnt2 + DemAnt3 * RetAnt3);
    Aux := Max(Aux,Max(DemResidual,0));
    end;
  DemResidual := Aux;
  DemResidualHm3 := PC.m3_Hm3_Intervalo(DemResidual);
  DemReservatorio := 0;
  for II := 1 to 3 do
  DemReservatorio := DemReservatorio + PCR.ObtemValorDemanda(dt, II, 'T');
  DemReservatorio := PCR.m3_Hm3_Intervalo(DemReservatorio);
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
  VolMetMax := PCR.VolumeMaximo*vVolMetMax.Get(Mes)/100;
  VolMetMin := PCR.VolumeMaximo*vVolMetMin.Get(Mes)/100;
  if VolMetMin < PCR.VolumeMinimo then VolMetMin := PCR.VolumeMinimo;
  CicloEvapo := 0;
  VolFinal := VolInicInterv;
while CicloEvapo <= 10 do
  begin
  VolMedio := (VolInicInterv + VolFinal)/2;
  AreaMedia := PCR.CalculaArea(VolMedio);
  AfluAux := 0.001 * AreaMedia * (PCR.ObtemPrecipitacaoUnitaria(dt)
             - PCR.ObtemEvaporacaoUnitaria(dt));
  VolAtual := VolInicInterv + AfluHm3 + VazaoPCsMontanteRes + AfluAux
              - DemReservatorio;
  if DemResidual <= 0 then
    begin
    if VolAtual <= VolMetMax then L := 0
    else
      begin
      Vertimento := (VolAtual - VolMetMax);
      L := PCR.Hm3_m3_Intervalo(Vertimento);
      end;
    end
  else
     if VolAtual < VolMetMin then L := 0
     else
       if (VolAtual - DemResidualHm3) < VolMetMin then
          L := PCR.Hm3_m3_Intervalo(VolAtual - VolMetMin)
       else
         if VolAtual <= VolMetMax then L := DemResidual
         else
           begin
           Vertimento := VolAtual - VolMetMax; // Possível vertimento
           if Vertimento < DemResidualHm3 then L := DemResidual
           else L := PCR.Hm3_m3_Intervalo(Vertimento);
           end;
  VolAtual := VolAtual - PCR.m3_Hm3_Intervalo(L);
  Difer := ABS(VolFinal - VolAtual);
  if Difer <= (0.01 * PCR.VolumeMaximo) then CicloEvapo := 11
  else CicloEvapo := CicloEvapo + 1;
  VolFinal := VolAtual;
  end;
PCR.AtribuiDefluvioPlanejado(dt, L);
  if dt = Projeto.Total_IntSim then
    begin
    GlobalObjects.Remove('vListaPCs');
    GlobalObjects.Remove('vVolMetMax');
    GlobalObjects.Remove('vVolMetMin');
    end;
end.
...
```

Anexo 1.5 ROTINA DE PLANEJAMENTO PARA A APLICAÇÃO DA REGRA DE ZONEAMENTO DE RESERVATÓRIO COM AVALIAÇÃO DE DEMANDAS

```
program PlanejaZoneamento;

var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
    // Declaração das variáveis auxiliares
    PCReserv   : string;  // Var. p/ conter o nome do primeiro PC da lista
    PCDemandas : string;  // Var. p/ conter o nome do último PC da lista
    vListaPCs  : object; // Vetor p/ conter os PCs do Reserv. até demandas
    N          : integer; // Número de PCs da lista
    I          : integer; // Var. p/ indexar a lista
    II         : integer; // Var. p/ indexar
    vVolume1 : object; // Vetor p/ conter os valores Volume1 do Zoneamento
    vVolume2 : object; // Vetor p/ conter os valores Volume2 do Zoneamento
    vVolume3 : object; // Vetor p/ conter os valores Volume3 do Zoneamento
    Volume1 : real; // Var. p/ conter o Parâmetro Volume1 p/ o cálculo
    Volume2 : real; // Var. p/ conter o Parâmetro Volume2 p/ o cálculo
    Volume3 : real; // Var. p/ conter o Parâmetro Volume3 p/ o cálculo
    Mes        : integer; // Var. p/ conter o mês referente ao dt atual
    Ano        : integer; // Var. p/ conter o ano referente ao dt atual
    PC         : object;  // Var. p/ conter o objeto PC (qualquer PC)
    PCAnt : object; // Var. p/ conter o objeto PC Anterior a qualquer PC
    PCR        : object;  // Var. p/ conter o objeto PCR (reservatório)
    PCMontante : object;  // Var. p/ conter o objeto PC de montante
    Plan : object; // Var. Planilha p/ conter dados de entrada do usuário
    dt         : integer; // Intervalo de Tempo
    DemReservatorio  : real; // Demandas do PC Reservatório
    Dem1, Dem2, Dem3 : real; // Demandas prim., sec. e terc. de cada PC
    DemAnt1, DemAnt2, DemAnt3 : real; // Demandas do PC Anterior (PCAnt)
    RetAnt1, RetAnt2, RetAnt3 : real; // Fração de retorno dessas demandas
    CoefPri, CoefSec, CoefTer : real; // Coef. de Racionamento
    SB : real; // Vazões afluentes das sub-bacias ao PC em estudo
    Aux : real; // Var. Auxiliar p/ conter demandas residuais
    DemResidual : real; // Demanda não atendida pelas bacias
    DemResidualHm3 : real; // Demanda não atendida pelas afluências em Hm3
    AfluHm3          : real; // Afluência total ao reservatório em Hm3
    VazaoPCsMontante : real; // Vazões de PCs de montante
    VolInicInterv    : real; // Volume do reservatório no início do dt
    VolDisp : real; // Volume disponível no reservatório no início do dt
    CapUtil : real; // Capacidade Util do Reservatório (VOLmax-VOLmin)
    Disp    : real; // Disponibilidade hídrica no intervalo de tempo
    L       : real; // Vazão de água a liberar do reservatório no dt
    Vertimento : real; // Var. p/ conter o Vertimento do Reservatório
    Int_Simul        : integer; // Número de intervalos de simulação
begin
  dt := Projeto.ObtemDeltaT; // Atribui Delta-T atual à variável dt
  Int_Simul := Projeto.Total_IntSim; // Número de intervalos de simulação
  if dt = 1 then
     begin
   // Dados do usuário - Via arq.EXCELL
     Plan := CreateObject(TPlanilha);
     Plan.LoadFromFile(' D:\Diretorio\Arquivo.xls ');
     PCReserv   := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
     PCDemandas := Plan.GetEntry(1,2); // Ret. string da L1 e C2 da Plan.
     vVolume1 := Plan.RowToVec(2,1,12); // Vetor com os valores Volume1
     vVolume2 := Plan.RowToVec(3,1,12); // Vetor com os valores Volume2
     vVolume3 := Plan.RowToVec(4,1,12); // Vetor com os valores Volume3
     // Criação da lista de PCs apartir dos pontos dados pelo usuário
     vListaPCs := Projeto.PCsEntreDois(PCReserv,PCDemandas);
```

```
     // Adiciona vetores à lista de variáveis globais
     GlobalObjects.Add('vListaPCs', vListaPCs);
     GlobalObjects.Add('vVolume1', vVolume1);
     GlobalObjects.Add('vVolume2', vVolume2);
     GlobalObjects.Add('vVolume3', vVolume3);
     FreeObject(Plan);
     end
  else
     begin  // Recupera vetores da lista de variáveis globais
     vListaPCs  := TStringList(GlobalObjects.Get('vListaPCs'));
     vVolume1    := TwsDFVec(GlobalObjects.Get('vVolume1'));
     vVolume2    := TwsDFVec(GlobalObjects.Get('vVolume2'));
     vVolume3    := TwsDFVec(GlobalObjects.Get('vVolume3'));
     end;
  PCR := TprPCP(vListaPCs.GetObject(0)); // Atribui o Reserv. à var. PCR
  if dt = 1 then VolInicInterv := PCR.VolumeInicial
            else VolInicInterv := PCR.ObtemVolume(dt-1);
// Cálculo da disponibilidade hídrica do reservatório
  VolDisp := VolInicInterv - PCR.VolumeMinimo;
  AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
  VazaoPCsMontante := 0;
  if PCR.PCs_aMontante > 0 then
     begin
     for II := 0 to (PCR.PCs_aMontante - 1) do
         begin
         PCMontante := PCR.PC_aMontante(II);
         Projeto.RealizaBalancoHidricoAte(PCMontante);
         VazaoPCsMontante := VazaoPCsMontante +
                             PCMontante.ObtemDefluencia(dt);
         end;
     VazaoPCsMontante := PCR.m3_Hm3_Intervalo(VazaoPCsMontante);
     end;
  Disp := VolDisp + AfluHm3 + VazaoPCsMontante;
  CapUtil := PCR.VolumeMaximo - PCR.VolumeMinimo; // Vol. útil do reserv.
// Cálculo da demanda do Reservatório
 DemReservatorio := 0;
 For II := 1 to 3 do
  DemReservatorio := DemReservatorio + PCR.ObtemValorDemanda(dt, II, 'T');
 DemReservatorio := PCR.m3_Hm3_Intervalo(DemReservatorio);
// Cálculo dos parâmetros Volume1, Volume2 e Volume3
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
  Volume1 := PCR.VolumeMaximo*(vVolume1.Get(Mes))/100 – PCR.VolumeMinimo;
  Volume2 := PCR.VolumeMaximo*(vVolume2.Get(Mes))/100 – PCR.VolumeMinimo;
  Volume3 := PCR.VolumeMaximo*(vVolume3.Get(Mes))/100 – PCR.VolumeMinimo;
  if (Disp – DemReservatorio) >= Volume2 then
     begin
     CoefPri := 1;
     CoefSec := 1;
     CoefTer := 1;
     end
  else
    if (Disp – DemReservatorio) >= Volume1 then
      begin
      CoefPri := 1;
      CoefSec := 1;
      CoefTer := 0;
      end
    else
      begin
      CoefPri := 1;
      CoefSec := 0;
      CoefTer := 0;
      end;
```

```
  N := vListaPCs.Count; // Obtem o número de elementos da lista de PCs
  Aux := 0; // Atribui primeiro valor à variável auxiliar
  for I := 1 to (N-1) do
    begin
    PC := TprPCP(vListaPCs.GetObject(I)); // Atribui PC índice I à var. PC
    Dem1 := CoefPri * (PC.ObtemValorDemanda(dt, 1, 'T'));
    PC.AtribuiValorDemanda(dt, 1, 'P', Dem1);
    Dem2 := CoefSec * (PC.ObtemValorDemanda(dt, 2, 'T'));
    PC.AtribuiValorDemanda(dt, 2, 'P', Dem2);
    Dem3 := CoefTer * (PC.ObtemValorDemanda(dt, 3, 'T'));
    PC.AtribuiValorDemanda(dt, 3, 'P', Dem3);
    SB   := PC.ObtemVazaoAfluenteSBs; // Vazão Afl. das Sub-bacias no dt
    PCAnt := TprPCP(vListaPCs.GetObject(I-1)); // PC índice I-1 em PCAnt
    DemAnt1 := CoefPri * (PCAnt.ObtemValorDemanda(dt, 1, 'T'));
    DemAnt2 := CoefSec * (PCAnt.ObtemValorDemanda(dt, 2, 'T'));
    DemAnt3 := CoefTer * (PCAnt.ObtemValorDemanda(dt, 3, 'T'));
    // Fatores de retorno das demandas do PC anterior
    RetAnt1 := PCAnt.FatorDeRetorno(1);
    RetAnt2 := PCAnt.FatorDeRetorno(2);
    RetAnt3 := PCAnt.FatorDeRetorno(3);
    VazaoPCsMontante := 0;
    if PC.PCs_aMontante > 1 then
       begin
         for II := 0 to (PC.PCs_aMontante - 1) do
           begin
           PCMontante := PC.PC_aMontante(II);
           if PCMontante <> PCAnt then
              begin
              Projeto.RealizaBalancoHidricoAte(PCMontante);
              VazaoPCsMontante := VazaoPCsMontante +
                                  PCMontante.ObtemDefluencia(dt);
              end;
           end;
       end;
    DemResidual := DemResidual + (Dem1 + Dem2 + Dem3) - SB –
                   VazaoPCsMontante - (DemAnt1 * RetAnt1 + DemAnt2 *
                   RetAnt2 + DemAnt3 * RetAnt3);
    Aux := Max(Aux,Max(DemResidual,0));
    end;
  // Cálculo da demanda residual
  DemResidual := Aux;
  DemResidualHm3 := PC.m3_Hm3_Intervalo(DemResidual);
{Quando a água geranda nas próprias sub-bacias atenderem as demandas o
valor de DemResidual será negativo. Então faz-se L = 0}
  if DemResidual <= 0 then
     begin
     if (Disp – DemReservatorio) <= Volume3 then L := 0
     else
       begin
       Vertimento := (Disp – DemReservatorio) – Volume3;
       L := PCR.Hm3_m3_Intervalo(Vertimento);
       end;
     end
  else //Se as sub-bacias não atendem às demandas, aplica-se o Zoneamento
    if (Disp – DemReservatorio) <= PCR.VolumeMinimo then L := 0
    else
      if (Disp – DemReservatorio) <= Volume3 then L := DemResidual
      else
        begin
        Vertimento := (Disp – DemReservatorio) – Volume3;
        if Vertimento < DemResidualHm3 then L := DemResidual
        else L := PCR.Hm3_m3_Intervalo(Vertimento); // Converte p/ m3
        end;
```

```
{O valor a ser liberado é atribuído à propriedade (variável)
DefluvioPlanejado no intervalo dt, ou seja, DeltaT atual}
  PCR.AtribuiDefluvioPlanejado(dt, L);

// Escrita do Arquivo LOG
  Saida.WriteLN('Intervalo         : ' + ValToStr(dt));
  Saida.WriteLN('Lista de PCs: ');
  Saida.WriteStrings(vListaPCs);
  Saida.WriteLN('Volume1       : ' + ValToStr(Volume1));
  Saida.WriteLN('Volume2       : ' + ValToStr(Volume2));
  Saida.WriteLN('Volume3       : ' + ValToStr(Volume3));
  Saida.WriteLN('Vz. Aflu.     : ' + ValToStr(PCR.ObtemVazaoAfluenteSBs));
  Saida.WriteLN('Disp          : ' + ValToStr(Disp));
  Saida.WriteLN('DemResiualdHm3: ' + ValToStr(DemResidualHm3));
  Saida.WriteLN('Vertimento    : ' + ValToStr(Vertimento));
  Saida.WriteLN('L             : ' + ValToStr(L));
  if dt = Projeto.Total_IntSim then
    begin
    GlobalObjects.Remove('vListaPCs');
    GlobalObjects.Remove('vVolume1');
    GlobalObjects.Remove('vVolume2');
    GlobalObjects.Remove('vVolume3');
    Saida.Show;
    end;
end.
```

Anexo 1.6 ROTINA DE PLANEJAMENTO QUE PROMOVE DEFLUÊNCIAS FIXAS PARA GERAÇÃO DE ENERGIA

```
program PlanejaDefluFixaEnergia;

var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
  // Declaração das variáveis auxiliares
  PCReserv : string;  // Var. p/ conter o nome do PC reservatório
  i        : integer; // Var. p/ indexar
  Mes      : integer; // Var. p/ conter o mês referente ao dt atual
  Ano      : integer; // Var. p/ conter o ano referente ao dt atual
  vCurvaDefluFixo : object; // Vetor p/ conter a curva de Dem. Energética
  DefluFixo : real; // Var. p/ conter o Deflúvio fixo p/ gerar energia
  PCR        : object;  // Var. p/ conter o objeto PCR (reservatório)
  AfluHm3    : real;    // Afluência total ao reservatório em Hm3
  Plan : object; // Var. Planilha p/ conter dados de entrada do usuário
  dt   : integer; // Intervalo de Tempo
  TotalDt    : integer; // Número de intervalos de simulação
  VolInicInterv : real; // Volume do reservatório no início do dt
  VolDisp : real; // Volume disponível no reservatório no início do dt
  CapUtil : real; // Capacidade Util do Reservatório (VOLmax-VOLmin)
  Disp    : real; // Disponibilidade hídrica no intervalo de tempo

begin
  dt := Projeto.ObtemDeltaT; // Atribui Delta-T atual à variável dt
  TotalDt := Projeto.Total_IntSim; // Número de intervalos de simulação
  if dt = 1 then
     begin
     // Entrada de dados do usuário, via arq.EXCELL
```

```
     Plan := CreateObject(TPlanilha);
     Plan.LoadFromFile('D:\Diretorio\Arquivo.xls');
     PCReserv := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
     vCurvaDefluFixo := Plan.RowToVec(3,1,12);
  // Criação da variável global vCurvaDefluFixo
     GlobalObjects.Add('vCurvaDefluFixo', vCurvaDefluFixo);
     FreeObject(Plan);
     end
  // Recupera dados da lista de variáveis globais
  else vCurvaDefluFixo := TwsDFVec(GlobalObjects.Get('vCurvaDefluFixo'));
  PCR := Projeto.PCPeloNome(PCReserv); // Atribui o Reserv. à var. PCR
  if dt = 1 then VolInicInterv := PCR.VolumeInicial
            else VolInicInterv := PCR.ObtemVolume(dt-1);
    // Calcula a disponibilidade hídrica do reservatório
  VolDisp := VolInicInterv - PCR.VolumeMinimo;
  AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
  Disp    := VolDisp + AfluHm3;
  CapUtil := PCR.VolumeMaximo - PCR.VolumeMinimo;
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
  DefluFixo := vCurvaDefluFixo.Get(Mes); // Le defluvio fixo do usuário
{O valor de vazao a ser liberado é atribuído à propriedade
DefluvioPlanejado no intervalo dt, ou seja, DeltaT atual}
  PCR.AtribuiDefluvioPlanejado(dt, DefluFixo);
// Escrita do Arquivo LOG
     Saida.WriteLN('PCReserv          : ' + PCReserv);
     Saida.WriteLN('Planeja-Intervalo : ' +
                   ValToStr(dt)+'/'+ValToStr(TotalDt));
     Saida.WriteLN('Dias no Intervalo : ' +
                   ValToStr(Projeto.DiasNoIntervalo));
     Saida.WriteLN('Vol.Inic. Interv. : ' + ValToStr(VolInicInterv));
     Saida.WriteLN('Vol. Disp.        : ' + ValToStr(VolDisp));
     Saida.WriteLN('Vazoes Afluentes  : ' +
                   ValToStr(PCR.ObtemVazaoAfluenteSBs));
     Saida.WriteLN('Vazoes Afl. (Hm3) : ' + ValToStr(AfluHm3));
     Saida.WriteLN('Disp. (Hm3)       : ' + ValToStr(Disp));
     Saida.WriteLN('Defluvio Fixo     : ' + ValToStr(DefluFixo));
  if dt = TotalDt then
    begin
    GlobalObjects.Remove('vCurvaDefluFixo');
    Saida.Show;
    end;
end.
```

Anexo 1.7 Rotina de Planejamento que Promove Defluências pela Avaliação das Demandas de Energia

```
program PlanejaAvaliaDemandas;

var Saida    : object; {Variável já inicializada pelo sistema}
    Projeto  : object; {Variável já inicializada pelo sistema}
  // Declaração das variáveis auxiliares
  PCReserv : string;  // Var. p/ conter o nome do PC reservatório
  i        : integer; // Var. p/ indexar
  dt       : integer; // Intervalo de Tempo
  TotalDt  : integer; // Número de intervalos de simulação
  Mes      : integer; // Var. p/ conter o mês referente ao dt atual
  Ano      : integer; // Var. p/ conter o ano referente ao dt atual
```

```
  DemandaEnergia : real;  // Var. p/ conter a Demanda energética
  Potencia       : real;  // Var. p/ cálculo da potência
  RendAducao     : real;  // Rendimento da adução
  RendTurbina    : real;  // Rendimento das turbinas
  RendGerador    : real;  // Rendimento dos geradores
  Queda          : real;  // Queda hidráulica
  PCR        : object;  // Var. p/ conter o objeto PCR (reservatório)
  AfluHm3    : real;    // Afluência total ao reservatório em Hm3
  VolInicInterv : real; // Volume do reservatório no início do dt
  VolDisp   : real; // Volume disponível no reservatório no início do dt
  Vazao     : real; // Vazao a ser turbinada no reservatório no dt
begin
  dt      := Projeto.ObtemDeltaT; // Atribui Delta-T atual à var. dt
  TotalDt := Projeto.Total_IntSim; // Número de intervalos de simulação
  PCReserv := 'Queimado';
  PCR := Projeto.PCPeloNome(PCReserv); // Atribui o Reserv. à var. PCR
  if dt = 1 then VolInicInterv := PCR.VolumeInicial
            else VolInicInterv := PCR.ObtemVolume(dt-1);
    // Cálculo da disponibilidade hídrica do reservatório
  VolDisp := VolInicInterv - PCR.VolumeMinimo;
  AfluHm3 := PCR.m3_Hm3_Intervalo(PCR.ObtemVazaoAfluenteSBs);
  Queda   := (PCR.CalculaCotaHidraulica(VolInicInterv) - PCR.CotaJusante);
  RendAducao  := PCR.RendimentoAducao;
  RendTurbina := PCR.RendimentoTurbina;
  RendGerador := PCR.RendimentoGeradores;
  Projeto.DeltaT_ComoData(dt, Mes, Ano); // Retorna mês e ano do dt atual
  DemandaEnergia := PCR.CurvaDemandaEnergetica(Mes); // Lê a Dem. Energia
  // Cálculo da vazão a ser liberada no reserv. p/ atender dem. energética
  Potencia := (DemandaEnergia / (Projeto.DiasNoIntervalo * 24));
  Vazao := Potencia / (9.81 * RendAducao * RendTurbina * RendGerador *
           Queda);
{O valor a ser liberado é atribuído à propriedade (variável)
DefluvioPlanejado no intervalo dt, ou seja, DeltaT atual}
  PCR.AtribuiDefluvioPlanejado(dt, Vazao);
  // Escrita do Arquivo LOG
  Saida.WriteLN('Planeja-Intervalo: '+ValToStr(dt)+'/'+ValToStr(TotalDt));
  Saida.WriteLN('Mês e Ano do Dt : '+ValToStr(Mes)+'/'+ValToStr(Ano));
  Saida.WriteLN('Nome do PC Reservatório: '+PCReserv);
  Saida.WriteLN('Dem. Energética no Mes : '+ValToStr(DemandaEnergia));
  Saida.WriteLN('Vol.Inicial do Interv. : '+ValToStr(VolInicInterv));
  Saida.WriteLN('Vol. Disponível        : '+ValToStr(VolDisp));
  Saida.WriteLN('Vazoes Afluentes (Hm3) : '+ValToStr(AfluHm3));
  Saida.WriteLN('Dias no Mês  : '+ValToStr(Projeto.DiasNoIntervalo));
  Saida.WriteLN('Horas no Mês : '+ValToStr(Projeto.DiasNoIntervalo * 24));
  Saida.WriteLN('Potência         : '+ValToStr(Potencia));
  Saida.WriteLN('Queda            : '+ValToStr(Queda));
  Saida.WriteLN('Vazao            : '+ValToStr(Vazao));
  if dt = TotalDt then
    begin
    GlobalObjects.Remove('vCurvaEnergia');
    Saida.Show;
    end;
end.
```

Anexo 1.8 ROTINA DE CÁLCULO DE ENERGIA DO USUÁRIO

```
Program EnergiaUsuario;
// Variáveis pré-declaradas
//   Saida   : TwsOutPut
//   Projeto : TprProjeto_Rosen
var
  dt       : Integer;
  Mes      : Integer;
  Ano      : Integer;
  IntTotal : Integer;
  i        : Integer;
  NLP      : Integer;
  NLV      : Integer;
  NomeArquivo : String;
  Planilha    : Object;
  NomePCR     : String;
  PCR         : Object;
  VolumeMaximo   : Real;
  VolumeMinimo   : Real;
  VolumeAtual    : Real;
  VolumeAnterior : Real;
  Vazao          : Real;
  VazaoMaxTurb   : Real;
  CotaMaxima     : Real;
  CotaMinima     : Real;
  CotaDoDt       : Real;
  CotaJusante    : Real;
  Queda          : Real;
  RendTurbina    : Real;
  RendGerador    : Real;
  RendAducao     : Real;
  Potencia        : Real;
  Energia         : Real;
  DemandaEnergia  : Real;
  CurvaDemEnergia : Real;
  // Variáveis usadas na interpolação
  A : Real;
  B : Real;
  C : Real;
  D : Real;
  E : Real;
  NomeSerie       : String;
  vCurvaVazao     : Object;
  vCurvaRendi     : Object;
  vAuxIndice      : Object;
  Indice          : Integer;

Begin
// Inialização dos intervalos de tempo
  dt := Projeto.ObtemDeltaT;
  IntTotal := Projeto.Total_IntSim;
  if dt = 1 then
    begin
      // Prepara planilha para receber dados de arquivo do usuário
      Planilha := CreateObject(TPlanilha);
      NomeArquivo := ' D:\Diretorio\Arquivo.xls';
      Planilha.LoadFromFile(NomeArquivo);
      NomePCR := Planilha.GetEntry(1, 1);
```

```
      // Lê dados do arquivo - N.linhas dos futuros vetores e da plan.
      NLV := Trunc(Planilha.GetFloat(1, 2));
      NLP := NLV + 2;
      NomeSerie := Planilha.GetEntry(2, 2);
      // Criação de Vetores
      vCurvaVazao := CreateObject(TwsDFVec, NLV);
      vCurvaRendi := CreateObject(TwsDFVec, NLV);
      //  Preechimento dos vetores a partir da planilha
      vCurvaVazao := Planilha.ColToVec(1, 3, NLP);
      vCurvaRendi := Planilha.ColToVec(2, 3, NLP);
      // Criação de variáveis globais (objetos)
      GlobalObjects.Add('vCurvaVazao', vCurvaVazao);
      GlobalObjects.Add('vCurvaRendi', vCurvaRendi);
      FreeObject(Planilha);
      end
    else
      begin
      // Recupera vetores dentre as variáveis globais
      vCurvaVazao := TwsDFVec(GlobalObjects.Get('vCurvaVazao'));
      vCurvaRendi := TwsDFVec(GlobalObjects.Get('vCurvaRendi'));
      end;
// Inicialização do PC reservatório;
  PCR := Projeto.PCPeloNome(NomePCR);
// Inicializando Volumes
  VolumeMaximo   := PCR.VolumeMaximo;
  VolumeMinimo   := PCR.VolumeMinimo;
  VolumeAtual    := PCR.ObtemVolume(dt);
  if dt = 1 then VolumeAnterior := PCR.VolumeInicial
     else VolumeAnterior := PCR.ObtemVolume(dt-1);
// Inicializando Cotas para cada volume do reservatório
  CotaMaxima  := PCR.CalculaCotaHidraulica(VolumeMaximo);
  CotaMinima  := PCR.CalculaCotaHidraulica(VolumeMinimo);
  CotaDoDt := PCR.CalculaCotaHidraulica((VolumeAtual+VolumeAnterior)/2);
  CotaJusante := PCR.CotaJusante;
// Inicializa queda hidráulica
  Queda := CotaDoDt - CotaJusante;
// Inicializa Vazão com dados do defluvio operado
  Vazao := PCR.ObtemDefluvioOperado(dt);
  Vazao := PCR.Hm3_m3_Intervalo(Vazao);
  VazaoMaxTurb := PCR.VazaoMaxTurbina;
  if Vazao > VazaoMaxTurb then Vazao := VazaoMaxTurb;
// Estabelece o rendimento da turbina conforme o valor de vazao
  B := (Vazao / VazaoMaxTurb) * 100;
  vAuxIndice := vCurvaVazao.FindGTE(B);
  Indice := vAuxIndice.GetAsInteger(1);
  FreeObject(vAuxIndice);
  if Indice = 1 then
    begin
      A := 0;
      D := 0;
    end
  else
    begin
      A := vCurvaVazao.Get(Indice-1);
      D := vCurvaRendi.Get(Indice-1);
    end;
  C := vCurvaVazao.Get(Indice);
  E := vCurvaRendi.Get(Indice);
  RendTurbina := ((E - D) * (Vazao - A) / (C - A) + D) / 100;
// Atribui demais rendimentos às variáveis
  RendGerador := PCR.RendimentoGeradores;
  RendAducao  := PCR.RendimentoAducao;
```

```
// Calcula Potência e Energia geradas
  Potencia := 9.81 * RendAducao * RendTurbina * RendGerador * Vazao *
              Queda;
  Energia := Potencia * (Projeto.DiasNoIntervalo * 24);
// Obtém valores das demandas de energia do PCR
  Projeto.DeltaT_ComoData(dt, Mes, Ano);
  DemandaEnergia  := PCR.DemandaEnergetica;
  CurvaDemEnergia := PCR.CurvaDemandaEnergetica(Mes);
// Atribui a energia calculada ao vetor de energia gerada
  PCR.AtribuiEnergia(dt, Energia);
// Apresentação dos Resultados em proces. de texto (log)
  Saida.WriteLN('Cota Max. : ' + ValToStr(CotaMaxima));
  Saida.WriteLN('Cota Do Dt: ' + ValToStr(CotaDoDt));
  Saida.WriteLN('Cota Min. : ' + ValToStr(CotaMinima));
  Saida.WriteLN('Cota Jus. : ' + ValToStr(CotaJusante));
  Saida.WriteLN('Queda     : ' + ValToStr(Queda));
  Saida.WriteLN('Vazao (m3/s)   : ' + ValToStr(Vazao));
  Saida.WriteLN('Vazao Max.Turb.: ' + ValToStr(VazaoMaxTurb));
  Saida.WriteLN('Rend. Turbina  : ' + ValToStr(RendTurbina));
  Saida.WriteLN('Rend. Aducao   : ' + ValToStr(RendAducao));
  Saida.WriteLN('Rend. Gerador  : ' + ValToStr(RendGerador));
  Saida.WriteLN('Potencia     : ' + ValToStr(Potencia));
  Saida.WriteLN('Dt (horas) : ' + ValToStr(Projeto.DiasNoIntervalo * 24));
  Saida.WriteLN('Energia Gerada   : ' + ValToStr(Energia));
  Saida.WriteLN('Demanda Energia  : ' + ValToStr(DemandaEnergia));
  Saida.WriteLN('Curva Dem.Energia: ' + ValToStr(CurvaDemEnergia));
// So mostra o editor no ultimo Delta T
  if dt = IntTotal then
    begin
      GlobalObjects.Remove('vCurvaVazao');
      GlobalObjects.Remove('vCurvaRendi');
      Saida.Show;
    end;
end.
```

Anexo 1.9 Rotina de Uso Geral para Determinar a Curva de Permanência da Energia Gerada

```
Program GeralCurvaPermanenciaEnergia;

// Variáveis pré-declaradas
//   Saida   : TwsOutPut
//   Projeto : TprProjeto_Rosen
var
  IntTotal : Integer;
  i        : Integer;
  PCR      : Object;
  Planilha1 : Object;
  Planilha2 : Object;
  Grafico   : Object;
  Serie1    : Object;
  Cor       : Integer;
  vEnergiaPCR     : Object;
  vEnergiaPCRDecr : Object;
  vFreqPCR        : Object;
  vTempo              : Object;
```

```
Begin
  // Inialização do intervalo total
  IntTotal := Projeto.Total_IntSim;
  // Inicialização dos reservatórios;
  PCR := Projeto.PCPeloNome('Queimado');
  // Criação e inicialização dos vetores
  vTempo          := CreateObject(TwsSIVec, IntTotal);
  vEnergiaPCR     := CreateObject(TwsDFVec, IntTotal);
  vEnergiaPCRDecr := CreateObject(TwsDFVec, IntTotal);
  vFreqPCR        := CreateObject(TwsDFVec, IntTotal);
  //  Preechimento dos vetores
  for i := 1 to IntTotal do
    begin
    vTempo.Set(i, i);
    vEnergiaPCR.Set(i, PCR.ObtemEnergia(i));
    end;
  // Cálculo da Curva de Permanência para Energia Gerada no PCR
  for i := 1 to IntTotal do
    vEnergiaPCRDecr.Set(i, vEnergiaPCR.Get(i));
  vEnergiaPCRDecr.Sort(false, false);
  for i := 1 to IntTotal do
    vFreqPCR.Set(i, (100*i/IntTotal));
  // Apresentação da Energia Gerada (em Planilha 1)
  Planilha1 := CreateObject(TPlanilha);
  Planilha1.Write(1, 1, 'Valores da Energia Gerada no Reservatório');
  Planilha1.Write(2, 1, 'Projeto ' + Projeto.Nome);
  Planilha1.Write(3, 1, 'Dt');
  Planilha1.Write(3, 2, 'PCR Queimado');
  Planilha1.WriteVecInCol(vTempo, 1, 4);
  Planilha1.WriteVecInCol(vEnergiaPCR, 2, 4);
  // Apresentação da Curva de Permanência da Energia Gerada (em Planilha2)
  Planilha2 := CreateObject(TPlanilha);
  Planilha2.Write(1, 1, 'Curva de Permanência da Energia Gerada no Res.');
  Planilha2.Write(2, 1, 'Projeto ' + Projeto.Nome);
  Planilha2.Write(3, 1, 'Dt');
  Planilha2.Write(3, 2, 'Energ.');
  Planilha2.Write(3, 3, 'Prob.');
  Planilha2.WriteVecInCol(vTempo, 1, 4);
  Planilha2.WriteVecInCol(vEnergiaPCRDecr, 2, 4);
  Planilha2.WriteVecInCol(vFreqPCR, 3, 4);
  // Monta o gráfico
  Cor := getColor('Black');
  Grafico := CreateObject(TgrGrafico);
  Grafico.Chart.SetView3D(False);
  Grafico.Chart.Title.Add('Curva de Permanência de Energia Gerada');
  Serie1 := Grafico.Series.AddLineSerie('PCR', Cor);
  for i := 1 to IntTotal do
    Serie1.AddXY(vFreqPCR.Get(i), vEnergiaPCRDecr.Get(i), '', Cor);
  // Mostra os dados em forma de planilha e gráfico
  Planilha1.Show;
  Planilha2.Show;
  Grafico.Show;
{ATENÇÃO: Os objetos Planilha, Grafico e Series (S1..S4) não deverão ser
destruídos pois continuarão a existirem após a execução de script
(mostrarão os dados) e serão destruídos automaticamente ao serem fechados
pelo usuário.}
end.
```

Anexo 1.10 Rotina de Uso Geral que Calcula e Apresenta a Curva de Energia Média Mensal Gerada e a Demanda Mensal

```
Program GeralCurvaEnergiaMedia;

// Variáveis pré-declaradas
//   Saida   : TwsOutPut
//   Projeto : TprProjeto_Rosen
var
  IntTotal  : Integer;
  i         : Integer;
  ii        : Integer;
  NovoValor : Real;
  PCR       : Object;
  Planilha : Object;
  Grafico  : Object;
  SerieDemandaEnergia  : Object;
  SerieEnergiaMedia    : Object;
  CorDemanda : Integer;
  CorEnergia : Integer;
  vTempo             : Object;
  vDemandaEnergiaPCR : Object;
  vEnergiaGeradaPCR  : Object;
  vEnergiaMediaPCR   : Object;
Begin
  // Inialização do intervalo total
  IntTotal := Projeto.Total_IntSim;
  // Inicialização dos reservatórios;
  PCR := Projeto.PCPeloNome('Queimado');
  // Criação e inicialização dos vetores
  vTempo := CreateObject(TwsSIVec, 12);
  vDemandaEnergiaPCR := CreateObject(TwsDFVec, 12);
  vEnergiaGeradaPCR  := CreateObject(TwsDFVec, IntTotal);
  vEnergiaMediaPCR   := CreateObject(TwsDFVec, 12);
  // Preechimento dos vetores de Demanda de Energia e Tempo
  for i := 1 to 12 do
     begin
     vTempo.Set(i, i);
     vDemandaEnergiaPCR.Set(i, PCR.CurvaDemandaEnergetica(i));
     end;
  // Preechimento do vetor de Energia Gerada
  for i := 1 to IntTotal do
    vEnergiaGeradaPCR.Set(i, PCR.ObtemEnergia(i));
  // Preechimento do vetor de Energia Média Gerada
  for i := 1 to 12 do
    for ii := 0 to (Projeto.NumAnosDeExecucao - 1) do
      begin
      NovoValor := vEnergiaGeradaPCR.Get(12*ii +i) /
                   Projeto.NumAnosDeExecucao;
      NovoValor := vEnergiaMediaPCR.Get(i) + NovoValor;
      vEnergiaMediaPCR.Set(i, NovoValor);
      end;
  // Apresentação da Energia Gerada em Planilha
  Planilha := CreateObject(TPlanilha);
  Planilha.Write(1, 1, 'Valores de Dem.Energética e Energia Gerada');
  Planilha.Write(2, 1, 'Mês');
  Planilha.Write(2, 2, 'Dem. Energia');
  Planilha.Write(2, 3, 'Energia Gerada');
  Planilha.WriteVecInCol(vTempo, 1, 3);
  Planilha.WriteVecInCol(vDemandaEnergiaPCR, 2, 3);
  Planilha.WriteVecInCol(vEnergiaMediaPCR, 3, 3);
```

```
  // Monta o gráfico
  CorDemanda := getColor('red');
  CorEnergia := getColor('blue');
  Grafico := CreateObject(TgrGrafico);
  Grafico.Chart.SetView3D(True);
  Grafico.Chart.Title.Add('Curva de Permanência de Energia Gerada');
  SerieDemandaEnergia := Grafico.Series.AddBarSerie('Dem.Energ.',
                         CorDemanda, 0,1);
  SerieEnergiaMedia := Grafico.Series.AddBarSerie('Ener. Gerada',
                       CorEnergia, 0,1);
  for i := 1 to 12 do
    begin
    SerieDemandaEnergia.AddXY(vTempo.Get(i), vDemandaEnergiaPCR.Get(i),
                              '', CorDemanda);
    SerieEnergiaMedia.AddXY(vTempo.Get(i), vEnergiaMediaPCR.Get(i),
                            '', CorEnergia);
    end;
  // Mostra os dados em forma de planilha e gráfico
  Planilha.Show;
  Grafico.Show;
// Liberação de objetos sem uso
  FreeObject(vEnergiaGeradaPCR);
  FreeObject(vTempo);
  FreeObject(vDemandaEnergiaPCR);
  FreeObject(vEnergiaMediaPCR);
  FreeObject(vDefluenciaPCR);
  FreeObject(vDefluPlanPCR);
  FreeObject(vDefluOperPCR);
{ATENÇÃO: Os objetos Planilha, Grafico e Series (S1..S4) não deverão ser
destruídos pois continuarão a existirem após a execução de script
(mostrarão os dados) e serão destruídos automaticamente ao serem fechados
pelo usuário.}
end.
```

Anexo 1.11 Rotina de Uso Geral Usada em Conjunto com a Rotina de Planejamento de Volumes-Metas

```
Program GeralVolumeMetas;
// Variáveis pré-declaradas
//   Saida : TwsOutPut    e   Projeto : TprProjeto_Rosen

var
  IntTotal  : Integer;
  i         : Integer;
  ii        : Integer;
  NovoValor : Real;
  PCR       : Object;
  PCReserv  : String;
  Plan     : Object;
  Planilha : Object;
  Grafico  : Object;
  SerieVolMetMax : Object;
  SerieVolMetMin : Object;
  SerieVolumesMedios : Object;
  CorMetaMax : Integer;
  CorMetaMin : Integer;
  CorVolume  : Integer;
```

```
  vMeses         : Object;
  vDeltaT        : Object;
  vVolMetMax     : Object;
  vVolMetMin     : Object;
  vVolumes       : Object;
  vVolumesMedios : Object;

Begin
  // Inialização do intervalo total
  IntTotal := Projeto.Total_IntSim;
  // Criação e inicialização dos vetores
  vMeses  := CreateObject(TwsSIVec, 12);
  vDeltaT := CreateObject(TwsDFVec, IntTotal);
  vVolMetMax  := CreateObject(TwsDFVec, 12);
  vVolMetMin  := CreateObject(TwsDFVec, 12);
  vVolumes       := CreateObject(TwsDFVec, IntTotal);
  vVolumesMedios := CreateObject(TwsDFVec, 12);
  // Preechimento do vetor Meses
  for i := 1 to 12 do
      vTempo.Set(i, i);
  // Entrada de dados do usuário, via arq.EXCELL
  Plan := CreateObject(TPlanilha);
  Plan.LoadFromFile(' D:\Diretorio\Arquivo.xls');
  PCReserv := Plan.GetEntry(1,1); // Ret. string da L1 e C1 da Plan.
  vVolMetMax := Plan.RowToVec(3,1,12); // Vetor com parâmetros Vol. Max.
  vVolMetMin := Plan.RowToVec(4,1,12); // Vetor com parâmetros Vol. Min.
  FreeObject(Plan);
  // Inicialização do nome do reservatório;
  PCR := Projeto.PCPeloNome(PCReserv);
  // Converte os valores dos vetores dos vol. metas em valores absolutos
  for i := 1 to 12 do
     begin
     vVolMetMax.Set(i, vVolMetMax.Get(i) * PCR.VolumeMaximo / 100);
     vVolMetMin.Set(i, vVolMetMin.Get(i) * PCR.VolumeMaximo / 100);
     end;
  // Preechimento do vetor de Volumes reais
  for i := 1 to IntTotal do
     begin
     vVolumes.Set(i, PCR.ObtemVolume(i));
     vDeltaT.Set(i, i);
     end;
  // Preechimento do vetor de Volumes Médios
  for i := 1 to 12 do
    for ii := 0 to (Projeto.NumAnosDeExecucao - 1) do
       begin
       NovoValor := vVolumes.Get(12*ii +i) / Projeto.NumAnosDeExecucao;
       NovoValor := vVolumesMedios.Get(i) + NovoValor;
       vVolumesMedios.Set(i, NovoValor);
       end;

  // Monta o gráfico
  CorMetaMin := getColor('green');
  CorVolume  := getColor('blue');
  CorMetaMax := getColor('red');
  Grafico := CreateObject(TgrGrafico);
  Grafico.Chart.SetView3D(True);
  Grafico.Chart.Title.Add('Volumes no Reservatório');
  SerieVolMetMin := Grafico.Series.AddBarSerie('Mínimo', CorMetaMin, 0,1);
  SerieVolumesMedios := Grafico.Series.AddBarSerie('Médios', CorVolume,
                        0,1);
  SerieVolMetMax := Grafico.Series.AddBarSerie('Máximo', CorMetaMax, 0,1);
```

```
  for i := 1 to 12 do
    begin
    SerieVolMetMin.AddXY(vTempo.Get(i), vVolMetMin.Get(i),
                         '', CorMetaMin);
    SerieVolumesMedios.AddXY(vTempo.Get(i), vVolumesMedios.Get(i),
                             '', CorVolume);
    SerieVolMetMax.AddXY(vTempo.Get(i), vVolMetMax.Get(i),
                         '', CorMetaMax);
    end;
// Mostra os dados em forma de gráfico
  Grafico.Show;
// Liberação de objetos sem uso
  FreeObject(vMeses);
  FreeObject(vDeltaT);
  FreeObject(vVolMetMax);
  FreeObject(vVolMetMin);
  FreeObject(vVolumesMedios);
  FreeObject(vVolumes);
{ATENÇÃO: Os objetos Planilha, Grafico e Series (S1..S4) não deverão ser
destruídos pois continuarão a existirem após a execução de script
(mostrarão os dados) e serão destruídos automaticamente ao serem fechados
pelo usuário.}
end.
```

Anexo 1.12 Rotina utilizada como "Script Geral" para Calcular uma Curva de Permanência de Vazões

```
Program ScriptGeralCurvaPermanenciaVazoes;

// Variáveis pré-declaradas
//   Saida   : TwsOutPut
//   Projeto : TprProjeto_Rosen

var
  NomeArquivo : String;
  NomeSerie   : String;
  NLP  : Integer;
  NLV  : Integer;
  i    : Integer;
  Planilha  : Object;
  Planilha1 : Object;
  Planilha2 : Object;
  Grafico   : Object;
  Serie1    : Object;
  Cor       : Integer;
  vIndiceSerie : Object;
  vSerie       : Object;
  vSerieDecr   : Object;
  vFreq        : Object;

Begin
  // Leitura de dados do arquivo do usuário
  NomeArquivo := ' D:\Diretorio\Arquivo.xls';
  Planilha := CreateObject(TPlanilha);
  Planilha1 := CreateObject(TPlanilha);
  Planilha2 := CreateObject(TPlanilha);
```

```
  Planilha.LoadFromFile(NomeArquivo);
  NLV := Trunc(Planilha.GetFloat(1, 2));
  NLP := NLV + 2;
  NomeSerie := Planilha.GetEntry(2, 2);
  // Criação de Vetores
  vIndiceSerie := CreateObject(TwsDFVec, NLV);
  vSerie       := CreateObject(TwsDFVec, NLV);
  vSerieDecr   := CreateObject(TwsDFVec, NLV);
  vFreq        := CreateObject(TwsDFVec, NLV);
  //  Preechimento dos vetores
  vIndiceSerie := Planilha.ColToVec(1, 3, NLP);
  vSerie       := Planilha.ColToVec(2, 3, NLP);
  // Cálculo da Curva de Permanência para Energia Gerada em Queimado
  for i := 1 to NLV do
    vSerieDecr.Set(i, vSerie.Get(i));
  vSerieDecr.Sort(false, false);
  for i := 1 to NLV do
    vFreq.Set(i, (100*i/NLV));
  // Apresentação da Energia Gerada (em Planilha 1)
  Planilha1.Write(1, 1, 'Valores da Energia Gerada nos Reservatórios');
  Planilha1.Write(2, 1, 'Projeto ' + Projeto.Nome);
  Planilha1.Write(3, 1, 'Num.');
  Planilha1.Write(3, 2, NomeSerie);
  Planilha1.WriteVecInCol(vIndiceSerie, 1, 4);
  Planilha1.WriteVecInCol(vSerie, 2, 4);
  // Apresentação da Curva de Permanência da Energia Gerada (em Planilha2)
  Planilha2.Write(1, 1, 'Curva de Permanência da Energia Gerada nos Res');
  Planilha2.Write(2, 1, 'Projeto ' + Projeto.Nome);
  Planilha2.Write(3, 1, 'Num.');
  Planilha2.Write(3, 2, NomeSerie);
  Planilha2.Write(3, 3, 'Prob.');
  Planilha2.WriteVecInCol(vIndiceSerie, 1, 4);
  Planilha2.WriteVecInCol(vSerieDecr, 2, 4);
  Planilha2.WriteVecInCol(vFreq, 3, 4);
  // Monta o gráfico
  Cor := getColor('TeeColor');
  Grafico := CreateObject(TgrGrafico);
  Grafico.Chart.SetView3D(False);
  Grafico.Chart.Title.Add('Curva de Permanência de ' + NomeSerie);
  Serie1 := Grafico.Series.AddLineSerie(NomeSerie, getColor('red'));
  for i := 1 to NLV do
    Serie1.AddXY(vFreq.Get(i), vSerieDecr.Get(i), '', Cor);
  // Mostra os dados em forma de planilha e gráfico
  Planilha1.Show;
  Planilha2.Show;
  Grafico.Show;
{ATENÇÃO: Os objetos Planilha, Grafico e Series (S1..S4) não deverão ser
destruídos pois continuarão a existirem após a execução de script
(mostrarão os dados) e serão destruídos automaticamente ao serem fechados
pelo usuário.}
end.
```

ANEXO 2 - PRINCIPAIS MÉTODOS DO PROPAGAR MOO

A seguir serão apresentados os principais métodos que compõem a estrutura do modelo Propagar MOO.

Anexo 2.1 MÉTODO SIMULACAO.

Esse método não foi alterado pelo presente trabalho. Ele é disparado automaticamente após o incremento do relógio de simulação e executa uma simulação na rede para o referido intervalo de simulação.

```
Procedure TprProjeto.Simulacao(Sender: TObject; const EventID: Integer);
var i: Integer;
begin
  if FdeltaT > FIntervalosSim then
     begin
     Fsimulador.Terminate;
     Exit;
     end;

  {O método a seguir faz o planejamento da distribuição de água em função
   de regras fornecidas pelo usuário no inicio de cada intervalo de tempo}
  Planeja; {metodo do projeto, devera ser interpretado ou padrao}

  {Dentro do intervalo de tempo todos os PCs são percorridos em ordem
   hierárquica para que se faça o Balanço Hídrico de cada um e o
   atendimento das Demandas}
  for i := 0 to FPCs.PCs-1 do
    FPCs.PC[i].BalancoHidrico;

  {O método a seguir tem por objetivo determinar um valor que sirva para
   avaliação da performance do sistema.}
   CalculaFuncaoObjetivo;

   // A informção a seguir possibilita em caso de interrupção informar
   // ao usuário em que ponto isso ocorreu.
   AtualizaPontoExecucao(‘’, nil);
end;
```

Anexo 2.2 MÉTODO PLANEJA.

Esse método não foi alterado pelo presente trabalho. Ele faz o planejamento da distribuição de água em função de regras fornecidas pelo usuário no início de cada intervalo de tempo.

```
procedure TprProjeto.Planeja;
var i: Integer;
    PC: TprPCP;
begin
  AtualizaPontoExecucao(FNome + '.Planeja', nil);

{Rotina míope - é o planejamento utilizado caso o usuário não programe
 nada. Para cada PC as Demandas Planejadas são definidas para atender as
 Demandas Totais solicitadas. }
  for i := 0 to FPCs.PCs-1 do
    begin
    PC := TprPCP(FPCs[i]);
    PC.DemPriPlanejada[FDeltaT] := PC.DemPriTotal[FDeltaT];
    PC.DemSecPlanejada[FDeltaT] := PC.DemSecTotal[FDeltaT];
    PC.DemTerPlanejada[FDeltaT] := PC.DemTerTotal[FDeltaT];

    // Caso o PC seja um Reservatório
    if PC is TprPCPR then
       TprPCPR(PC).DefluvioPlanejado[FDeltaT] := 0;
    end;

  // Executa o script Planeja do usuário caso exista.
  if FPlanejaUsuario <> '' then
     begin
     AtualizaPontoExecucao(FNome + '.ScriptPlaneja', FPlanejaScript);
     FPlanejaScript.Execute;
     AtualizaPontoExecucao('', nil);
     end;
end;
```

Anexo 2.3 MÉTODO OPERA.

Esse método também não foi alterado no presente trabalho. O método opera somente é executado em PCs com controle de reservatório.

```
procedure TprPCPR.Opera(
                { passagem por valor}
                const VolumeDisponivel, DPP, DSP, DTP, Deflu_Pl: Real;
                { variaveis de saída }
                out DPO, DSO, DTO, Deflu_OP: Real);

    { DPP      = Demanda Primária Planejada
      DSP      = Demanda Secundária Planejada
      DST      = Demanda Terciária Planejada
      Deflu_Pl = Defluvio Planejado
      DPO      = Demanda Primária Operada
      DSO      = Demanda Secundária Operada
      DTO      = Demanda Terciária Operada
      Deflu_OP = Defluvio Operado              }

    {IMPORTANTE: Deflu_Op não deve incorporar os RETORNOS das
     Demandas Operadas no Reservatorio - estes são considerados
     como que retornando ao curso d'água à jusante do mesmo.}
var i: Integer;
    PC: TprPCP;
begin
  FExecutarOperaPadrao := True;
  if FExecutarOperaPadrao then // Operação míope
     begin
     DPO      := DPP;
     DSO      := DSP;
     DTO      := DTP;
     Deflu_OP := Deflu_Pl;
     end;
  if FOperaScript <> nil then
     begin
     Projeto.AtualizaPontoExecucao(FNome + '.ScriptOpera', FOperaScript);
     // Script do usuário
     FOperaScript.Execute;
     Projeto.AtualizaPontoExecucao(FNome + '.Opera', nil);
     end;
end;
```

Anexo 2.4 MÉTODO BALANÇO HÍDRICO DE PC SEM RESERVATÓRIO.

Embora esse método tenha sido alterado no presente trabalho, aqui ele é apresentado na sua forma original, ou seja, não contendo as alterações propostas. O método modificado é apresentado no Anexo 3 desse trabalho.

```
{ Realiza o balanço Hídricö: X[t] = Q[t] - D[t] + RD[t]
Atenção: Teoricamente o método PrepararParaSimulacao já foi disparado.}
procedure TprPCP.BalancoHidrico;
var   i              : Integer;  // Intervalo de Simulação
      j              : Integer;  // Auxiliar
      DemandaTotal   : Real;     // Soma das demandas Totais
      AfluenciaTotal : Real;     // Toda a água que vem de trás deste PC
      DPP            : Real;     // Aux. - Demanda Primaria Planejada
      DSP            : Real;     // Aux. - Demanda Secundaria Planejada
      DTP            : Real;     // Aux. - Demanda Terciaria Planejada
      DPA            : Real;     // Aux. - Demanda Primaria Atendida
      DSA            : Real;     // Aux. - Demanda Secundaria Atendida
      DTA            : Real;     // Aux. - Demanda Terciaria Atendida
      Retorno        : Real;     // Aux. - Fração de Retorno
Begin
  Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
  i := FProjeto.DeltaT;
  // Determina a vazão afluente das Sub-bacias que afluem ao PC
FAfluenciaSB[i] := ObtemVazaoAfluenteSBs;
  FVzMon[i] := ObtemVazoesDeMontante;
  AfluenciaTotal := M3_Hm3_Intervalo(FVzMon[i]);
  // Totalização da agua afluente ao PC no DeltaT atual
  // FAfluenciaSB já esta sendo inicializado em PrepararParaSimulacao
  AfluenciaTotal := AfluenciaTotal + M3_Hm3_Intervalo(FAfluenciaSB[i]);
  // Análise para atendimento das Demandas Hídricas no PC.
  DPP := M3_Hm3_Intervalo(FDPP[i]);
  DSP := M3_Hm3_Intervalo(FDSP[i]);
  DTP := M3_Hm3_Intervalo(FDTP[i]);
  DemandaTotal := DPP + DSP + DTP;
  if AfluenciaTotal > DemandaTotal then
     begin
     FDPA[i] := Hm3_m3_Intervalo(DPP);
     FDSA[i] := Hm3_m3_Intervalo(DSP);
     FDTA[i] := Hm3_m3_Intervalo(DTP);
     Retorno := FFRP*DPP + FFRS*DSP + FFRT*DTP;
     end
  else
     begin
     // Não existe água para atender todas as Demandas:
     // fazer racionamento - chama método Raciona
     Raciona(AfluenciaTotal, DPP, DSP, DTP, {out} DPA, DSA, DTA);
     Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
     FDPA[i] := Hm3_m3_Intervalo(DPA);
     FDSA[i] := Hm3_m3_Intervalo(DSA);
     FDTA[i] := Hm3_m3_Intervalo(DTA);
     DemandaTotal := DPA + DSA + DTA;
     Retorno := FFRP*DPA + FFRS*DSA + FFRT*DTA;
     end;
  FDefluencia[i] := Hm3_m3_Intervalo (AfluenciaTotal - DemandaTotal +
Retorno);
end;
```

Anexo 2.5 MÉTODO BALANÇO HÍDRICO DE PC COM RESERVATÓRIO.

Esse método também foi alterado no presente trabalho, mas aqui ele é apresentado na sua forma original, ou seja, não contendo as alterações propostas.

```
procedure TprPCPR.BalancoHidrico;
Var i               : Integer;    // Intervalo de Simulação Atual
    j               : Integer;    // Auxiliar
    DemandaTotal    : Real;       // Soma das demandas Totais
    AfluenciaTotal  : Real;       // Água que aflui ao PC no Intervalo
    AfluenciaAux    : Real;       // Auxiliar
    PriVez          : Boolean;    // Auxiliar
    VolumeAtual     : Real;       // Auxiliar
    VolumeAux       : Real;       // Auxiliar
    AguaDisponivel  : Real;       // Auxiliar
    VolInicioInterv : Real;       // Auxiliar
    VolFinalInterv  : Real;       // Auxiliar
    VolumeMedio     : Real;       // Auxiliar
    AreaMedia       : Real;       // Auxiliar
    EvapoMedia      : Real;       // Auxiliar
    EvapoUnitaria   : Real;       // Auxiliar
    PrecipUnitaria  : Real;       // Auxiliar
    CicloEvapo      : Integer;    // Auxiliar
    DPT             : Real;       // Aux. - Demanda Primaria Total
    DST             : Real;       // Aux. - Demanda Secundaria Total
    DTT             : Real;       // Aux. - Demanda Terciaria Total
    DPP             : Real;       // Aux. - Demanda Primaria Planejada
    DSP             : Real;       // Aux. - Demanda Secundaria Planejada
    DTP             : Real;       // Aux. - Demanda Terciaria Planejada
    DPA             : Real;       // Aux. - Demanda Primaria Atendida
    DSA             : Real;       // Aux. - Demanda Secundaria Atendida
    DTA             : Real;       // Aux. - Demanda Terciaria Atendida
    DPO             : Real;       // Aux. - Demanda Primaria Operada
    DSO             : Real;       // Aux. - Demanda Secundaria Operada
    DTO             : Real;       // Aux. - Demanda Terciaria Operada
    Deflu_OP        : Real;       // Aux. - Defluvio Operado
    Deflu_Pl        : Real;       // Aux. - Defluvio Planejado
    Sair            : Boolean;
    Retorno         : Real;       // Aux. - Fração de Retorno
    Queda           : Real;       // Aux. - Queda Hidráulica
    Vazao           : Real;       // Aux. -  em m3/s
Begin
  Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
  i               := FProjeto.DeltaT;
  PriVez          := True;
  VolumeAtual     := 0;
  VolInicioInterv := ObtemVolFinalIntervaloAnterior;
  VolFinalInterv  := VolInicioInterv;
  EvapoMedia      := 0;
  EvapoUnitaria   := EvaporacaoUnitaria[i];
  PrecipUnitaria  := PrecipitacaoUnitaria[i];
  CicloEvapo      := 1;
  DPO             := 0;
  DSO             := 0;
  DTO             := 0;
  Deflu_OP        := 0;
  DPT             := M3_Hm3_Intervalo(FDPT[i]);
  DST             := M3_Hm3_Intervalo(FDST[i]);
  DTT             := M3_Hm3_Intervalo(FDTT[i]);
```

```
  {Calculo das vazões afluentes das Sub-bacias do PC e dos PCs à montante
   do Reservatorio
  Cálculo do campo AfluenciaSB FAfluenciaSB[i] é iniciada em zero por
  método da própria Classe SubBacia }
  FAfluenciaSB[i] := ObtemVazaoAfluenteSBs;
  FVzMon[i] := ObtemVazoesDeMontante;
  AfluenciaTotal := M3_Hm3_Intervalo(FVzMon[i]);
  // Totalização da agua afluente ao PC no DeltaT atual
  // FAfluenciaSB já esta sendo inicializado em PrepararParaSimulacao
  AfluenciaTotal := AfluenciaTotal + M3_Hm3_Intervalo(FAfluenciaSB[i]);
  DPP := M3_Hm3_Intervalo(FDPP[i]);
  DSP := M3_Hm3_Intervalo(FDSP[i]);
  DTP := M3_Hm3_Intervalo(FDTP[i]);
  Deflu_Pl := M3_Hm3_Intervalo(FDeflu_Pl[i]);
{ Inicio da OPERACAO DO RESERVATORIO: A operação é feita dentro de um
  conjunto de CICLOS DE AJUSTE DA EVAPORAÇÃO DO RESERVATORIO }
  CicloEvapo := 0;
  Sair := False;
  Repeat
    inc(CicloEvapo);
    VolumeMedio := (VolInicioInterv + VolFinalInterv) / 2;
    AreaMedia   := CalculaAreaDoReservatorio(VolumeMedio);
    // Retira da AfluenciaTotal a EvaporacaoMedia menos o que precipitou
    // sobre o reservatório no intervalo.
    // A multiplicação abaixo por 0,001 transforma mm*km2 em Hm3
    AfluenciaAux := AfluenciaTotal - 0.001 * AreaMedia * (EvapoUnitaria –
    PrecipUnitaria);
    // VERIFICA SE EXISTE ÁGUA suficiente para prosseguir o processamento:
       // se o Armazenamento Inicial é zero e se não existe
       // afluência, portanto, não existe água para o atendimento
       // de nenhuma demanda no PC.
    if (VolInicioInterv = 0) and (AfluenciaAux <= 0) then Break;
    // Se o armazenamento no inicio do mes nao é nulo, pode acontecer
    // que a afluencia seja negativa. Isso acontece quando nao existe
    // afluencia de fato, so evaporacao. Assim o proximo passo e
    // testar se isso nao ira zerar a existencia de agua ou, entao, a
    // torne tao pequena (menor que o Volume Morto = VolumeMinimo) que
    // nao possa satisfazer as demandas.
    VolumeAtual := VolInicioInterv + AfluenciaAux;
    if VolumeAtual <= 0 then
      // É possível que em uma primeira vez isso aconteça em função de
      // uma evaporação excessiva em virtude de uma aproximação
      // grosseira no cálculo. Assim é tentada uma segunda vez agora com
      // o VolFinalInterv = 0. Isso é controlado pela variável auxiliar
      // PrimVez.
      if Not PriVez then
         begin
         VolumeAtual:= 0;
         VolFinalInterv := VolumeAtual;
         Break; // NÃO HÁ ÁGUA - Sai do ciclo
         End
      Else
         begin
         VolumeAtual:= 0;
         VolFinalInterv := 0;
         PriVez := False;
         Continue;
         end
    else
      PriVez := True;
    // ... continua o teste da disponibilidade de água; agora se o
    // VolumeAtual nao é inferior ao VolumeMinimo.
    if VolumeAtual <= VolumeMinimo then
```

```
         // Mais uma vez é possível que isso aconteça em função de uma
         // evaporação excessiva em virtude de uma aproximação grosseira no
         // cálculo. Assim é tentada uma segunda vez agora com o
         // o VolFinalInterv = VolumeAtual. Isso é controlado, também, pela
         // variável auxiliar PrimVez.
         if Not PriVez then
            begin
            VolFinalInterv := VolumeAtual;
            Break; // NÃO HÁ ÁGUA - Sai do ciclo
            end
         else
            begin
            VolFinalInterv := VolumeAtual;
            PriVez := False;
            Continue;
            end
      else
        PriVez := True;
      {EXISTE ÁGUA - é chamado o método Opera que contém as regras de
       operação do reservatório. Este método pode ser escrito pelo usuário
       com o propósito de testar regras operacionais. Seu funcionamento pode
       ser combinado com o método Planeja da Classe Projeto.
       É feita a verificação se as demandas planejadas podem ser atendidas
       totalmente ou apenas em parte.
       Como saída este método dá:
         DPO = Demanda Primaria Operada,
         DSO = Demanda Secundária Operada,
         DTO = Demanda Terciária Operada e
         Deflu_OP = defluvio operado (agua que é liberada para jusante)
                    IMPORTANTE: Deflu_Op não deve incorporar os RETORNOS das
                                Demandas Operadas no Reservatorio - estes são
                                 considerados como que retornando ao curso
                                 d'água à jusante do mesmo.
       Outras variaveis podem ser alteradas (ver cabeçalho do método) sob
       exclusiva responsabilidade do usuário.}
      VolumeAux := VolumeAtual;
      Opera(VolumeAtual, DPP, DSP, DTP, Deflu_Pl, {out} DPO, DSO, DTO,
            Deflu_OP);
Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
      // Verificação da CONSISTÊNCIA das DEMANDAS OPERADAS.
      if DPO > DPT then
         begin
         DPO := DPT;
         // avisa do erro
         end;
      if DSO > DST then
         begin
         DSO := DST;
         // avisa do erro
         end;
      if DTO > DTT then
         begin
         DTO := DTT;
         // avisa do erro
         end;
      // Verifica a CONSISTÊNCIA de saída da OPERA em termos de
      // VIABILIDADE FÍSICA DO VOLUME DO RESERVATÓRIO
      VolumeAtual := VolumeAtual - (DPO + DSO + DTO + Deflu_OP);
      // verifica se o VolumeAtual é inferior ao mínimo
      if VolumeAtual < VolumeMinimo then
         begin
         // procura atender à limitação do VolumeMinimo reduzindo a
         // defluência operada.
```

```
        Deflu_OP := Deflu_OP - (VolumeMinimo - VolumeAtual);
        if Deflu_OP >= 0 then
           // somente a redução da defluência foi suficiente
           // DPO, DSO e DTO continuam atendidas conforme a OPERA
           VolumeAtual := VolumeMinimo
        else
           {NÃO É SUFICIENTE REDUZIR APENAS A DEFLUÊNCIA: esta situação já
            deveria ter sido tratada em Planeja e/ou em Opera através de um
            critério de racionamento de modo que agora não acontecesse esse
            tipo de inviabilidade. Entretanto, esses testes são feitos
            exatamente para que se isso venha a acontecer, não tendo sido
            tratado convenientemente pelo usuário não prejudique o
            desenvolvimento da simulação.}
           begin
             // existe água a ser racionada - é chamado o RACIONA
             // a saída da OPERA é tornada sem efeito
             // o VolumeAtual é reconstituído para o valor anterior à
             // utilização da OPERA
             VolumeAtual := VolumeAux;
             AguaDisponivel := VolumeAtual - VolumeMinimo;
             Raciona(AguaDisponivel, DPP, DSP, DTP, {out} DPA, DSA, DTA);
             Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
             DPO         := DPA;
             DSO         := DSA;
             DTO         := DTA;
             Deflu_OP    := 0 ;
             VolumeAtual := VolumeMinimo;
           end;
        end; // VolumeAtual é maior que VolumeMinimo ...
      // verifica se o VolumeAtual é superior ao máximo
      if VolumeAtual > VolumeMaximo then
         begin
         Deflu_Op := Deflu_Op + (VolumeAtual - VolumeMaximo);
         VolumeAtual := VolumeMaximo;
         end;
      // verifica CONVERGÊNCIA de VOLUMES para cálculo da EVAPORAÇÃO
      Sair := (ABS(VolFinalInterv - VolumeAtual) <= (0.01 * VolumeMaximo));
      VolFinalInterv := VolumeAtual;
    until Sair or (CicloEvapo > 100);
    // Atualização das Demandas Atendidas.
    FDPA[I]    := Hm3_m3_Intervalo(DPO);
    FDSA[I]    := Hm3_m3_Intervalo(DSO);
    FDTA[I]    := Hm3_m3_Intervalo(DTO);
    FVolume[i] := VolFinalInterv;
    Retorno    := FFRP*DPO + FFRS*DSO + FFRT*DTO;
    // Atualização da Defluencia.
    FDefluencia[i]:= Hm3_m3_Intervalo(Deflu_OP + Retorno);
    // Cálculo da Energia Gerada quando houver geração no Reservatorio
    if FDPE then
       begin
       Queda := CalculaQuedaHidraulica(VolumeAtual);
       Vazao := Hm3_m3_Intervalo(Deflu_OP);
       FEG[i] := CalculaEnergia(Queda, Vazao);
       end;
end; // PCPR.BalancoHidrico
```

ANEXO 3 - MÉTODOS RESPONSÁVEIS PELA SIMULAÇÃO DA GERAÇÃO DE ENERGIA NO MODELO PROPAGAR MOO

O cálculo da geração de energia no modelo Propagar MOO ocorre dentro dos métodos que realizam o balanço hídrico de PCs com e sem o controle de reservatório. Os referidos métodos contendo as alterações propostas nesse trabalho são apresentados a seguir.

Anexo 3.1 MÉTODO BALANÇO HÍDRICO DE PC SEM RESERVATÓRIO.

```
procedure TprPCP.BalancoHidrico;
var   i              : Integer;  // Intervalo de Simulação
      j              : Integer;  // Auxiliar
      DemandaTotal   : Real;     // Soma das demandas Totais
      AfluenciaTotal : Real;     // Toda a água que vem de trás deste PC
      DPP            : Real;     // Aux. - Demanda Primaria Planejada
      DSP            : Real;     // Aux. - Demanda Secundaria Planejada
      DTP            : Real;     // Aux. - Demanda Terciaria Planejada
      DPA            : Real;     // Aux. - Demanda Primaria Atendida
      DSA            : Real;     // Aux. - Demanda Secundaria Atendida
      DTA            : Real;     // Aux. - Demanda Terciaria Atendida
      Retorno        : Real;     // Aux. - Fração de Retorno
begin
  Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
  i := FProjeto.DeltaT;
  FAfluenciaSB[i] := ObtemVazaoAfluenteSBs;
  FVzMon[i] := ObtemVazoesDeMontante;
  AfluenciaTotal := M3_Hm3_Intervalo(FVzMon[i]);
  AfluenciaTotal := AfluenciaTotal + M3_Hm3_Intervalo(FAfluenciaSB[i]);
  DPP := M3_Hm3_Intervalo(FDPP[i]);
  DSP := M3_Hm3_Intervalo(FDSP[i]);
  DTP := M3_Hm3_Intervalo(FDTP[i]);
  DemandaTotal := DPP + DSP + DTP;
  if AfluenciaTotal > DemandaTotal then
     begin
     FDPA[i] := Hm3_m3_Intervalo(DPP);
     FDSA[i] := Hm3_m3_Intervalo(DSP);
     FDTA[i] := Hm3_m3_Intervalo(DTP);
     Retorno := FFRP*DPP + FFRS*DSP + FFRT*DTP;
     end
  else
     begin
     Raciona(AfluenciaTotal, DPP, DSP, DTP, {out} DPA, DSA, DTA);
     Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
     FDPA[i] := Hm3_m3_Intervalo(DPA);
     FDSA[i] := Hm3_m3_Intervalo(DSA);
     FDTA[i] := Hm3_m3_Intervalo(DTA);
     DemandaTotal := DPA + DSA + DTA;
     Retorno := FFRP*DPA + FFRS*DSA + FFRT*DTA;
     end;
  FDefluencia[i] := Hm3_m3_Intervalo (AfluenciaTotal - DemandaTotal +
                    Retorno);
if FGerarEnergia then
  begin
  FEG[i] := CalculaEnergia(FQuedaFixa, FDefluencia[i]);
  if FEnergiaScript <> nil then
   begin
 FProjeto.AtualizaPontoExecucao(FNome + '.ScriptEnergia', FEnergiaScript);
   FEnergiaScript.Execute;
   end;
  end;
end;
```

Anexo 3.2 MÉTODO BALANÇO HÍDRICO DE PC COM RESERVATÓRIO.

```
procedure TprPCPR.BalancoHidrico;
Var i               : Integer;    // Intervalo de Simulação Atual
    j               : Integer;    // Auxiliar
    DemandaTotal    : Real;       // Soma das demandas Totais
    AfluenciaTotal  : Real; // Toda a água que aflui ao PC no Intervalo
    AfluenciaAux    : Real;       // Auxiliar
    PriVez          : Boolean;    // Auxiliar
    VolumeAtual     : Real;       // Auxiliar
    VolumeAux       : Real;       // Auxiliar
    AguaDisponivel  : Real;       // Auxiliar
    VolInicioInterv : Real;       // Auxiliar
    VolFinalInterv  : Real;       // Auxiliar
    VolumeMedio     : Real;       // Auxiliar
    AreaMedia       : Real;       // Auxiliar
    EvapoMedia      : Real;       // Auxiliar
    EvapoUnitaria   : Real;       // Auxiliar
    PrecipUnitaria  : Real;       // Auxiliar
    CicloEvapo      : Integer;    // Auxiliar
    DPT             : Real;       // Aux. - Demanda Primaria Total
    DST             : Real;       // Aux. - Demanda Secundaria Total
    DTT             : Real;       // Aux. - Demanda Terciaria Total
    DPP             : Real;       // Aux. - Demanda Primaria Planejada
    DSP             : Real;       // Aux. - Demanda Secundaria Planejada
    DTP             : Real;       // Aux. - Demanda Terciaria Planejada
    DPA             : Real;       // Aux. - Demanda Primaria Atendida
    DSA             : Real;       // Aux. - Demanda Secundaria Atendida
    DTA             : Real;       // Aux. - Demanda Terciaria Atendida
    DPO             : Real;       // Aux. - Demanda Primaria Operada
    DSO             : Real;       // Aux. - Demanda Secundaria Operada
    DTO             : Real;       // Aux. - Demanda Terciaria Operada
    Deflu_OP        : Real;       // Aux. - Defluvio Operado
    Deflu_Pl        : Real;       // Aux. - Defluvio Planejado
    Sair            : Boolean;
    Retorno         : Real;       // Aux. - Fração de Retorno
    Queda           : Real;       // Aux. - Queda Hidráulica
    Vazao           : Real;       // Aux. -  em m3/s
    Vertimento      : Real;       // Aux.
Begin
  Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);

  // Inicializa variáveis auxiliares
  i               := FProjeto.DeltaT;
  PriVez          := True;
  VolumeAtual     := 0;
  VolInicioInterv := ObtemVolFinalIntervaloAnterior;
  VolFinalInterv  := VolInicioInterv;
  EvapoMedia      := 0;
  EvapoUnitaria   := EvaporacaoUnitaria[i];
  PrecipUnitaria  := PrecipitacaoUnitaria[i];
  CicloEvapo      := 1;
  DPO             := 0;
  DSO             := 0;
  DTO             := 0;
  DPT             := M3_Hm3_Intervalo(FDPT[i]);
  DST             := M3_Hm3_Intervalo(FDST[i]);
  DTT             := M3_Hm3_Intervalo(FDTT[i]);
  Vertimento      := 0;
  Deflu_OP        := 0;
```

```
  {Calculo das vazões afluentes - das Sub-bacias do PC e - dos PCs à
   montante do Reservatorio
   Cálculo do campo AfluenciaSB - FAfluenciaSB[i] é iniciada em zero por
   método da própria Classe SubBacia }

  FAfluenciaSB[i] := ObtemVazaoAfluenteSBs;
  FVzMon[i] := ObtemVazoesDeMontante;
  AfluenciaTotal := M3_Hm3_Intervalo(FVzMon[i]);
// FAfluenciaSB já esta sendo inicializado em PrepararParaSimulacao
  AfluenciaTotal := AfluenciaTotal + M3_Hm3_Intervalo(FAfluenciaSB[i]);
// Prepara para análise para atendimento das Demandas Hídricas no PC.
  DPP := M3_Hm3_Intervalo(FDPP[i]);
  DSP := M3_Hm3_Intervalo(FDSP[i]);
  DTP := M3_Hm3_Intervalo(FDTP[i]);
  Deflu_Pl := M3_Hm3_Intervalo(FDeflu_Pl[i]);

{ Inicio da OPERACAO DO RESERVATORIO: A operação é feita dentro de um
  conjunto de CICLOS DE AJUSTE DA EVAPORAÇÃO DO RESERVATORIO }
  CicloEvapo := 0;
  Sair := False;
  Repeat
    inc(CicloEvapo);
    VolumeMedio := (VolInicioInterv + VolFinalInterv) / 2;
    AreaMedia   := CalculaAreaDoReservatorio(VolumeMedio);
    // Retira da AfluenciaTotal a EvaporacaoMedia menos o que precipitou
    // sobre o reservatório no intervalo.
    // A multiplicação abaixo por 0,001 transforma mm*km2 em Hm3
    AfluenciaAux := AfluenciaTotal - 0.001 * AreaMedia * (EvapoUnitaria -
PrecipUnitaria);
    // VERIFICA SE EXISTE ÁGUA suficiente para prosseguir o processamento:
       // se o Armazenamento Inicial é zero e se não existe
       // afluência, portanto, não existe água para o atendimento
       // de nenhuma demanda no PC.
    if (VolInicioInterv = 0) and (AfluenciaAux <= 0) then Break;
    // Se o armazenamento no inicio do mes nao é nulo, pode acontecer
    // que a afluencia seja negativa. Isso acontece quando nao existe
    // afluencia de fato, so evaporacao. Assim o proximo passo e
    // testar se isso nao ira zerar a existencia de agua ou, entao, a
    // torne tao pequena (menor que o Volume Morto = VolumeMinimo) que
    // nao possa satisfazer as demandas.
    VolumeAtual := VolInicioInterv + AfluenciaAux;
    if VolumeAtual <= 0 then
      // É possível que em uma primeira vez isso aconteça em função de
      // uma evaporação excessiva em virtude de uma aproximação
      // grosseira no cálculo. Assim é tentada uma segunda vez agora com
      // o VolFinalInterv = 0. Isso é controlado pela variável auxiliar
      // PrimVez.
      begin
      VolumeAtual:= 0;
      if Not PriVez then
         begin
         VolFinalInterv := VolumeAtual;
         Break; // NÃO HÁ ÁGUA - Sai do ciclo
         end
      else
         begin
         VolFinalInterv := 0;
         PriVez := False;
         Continue;
         end
      end
    else
      PriVez := True;
```

```
  // ... continua o teste da disponibilidade de água; agora se o
  // VolumeAtual nao é inferior ao VolumeMinimo.
  if VolumeAtual <= VolumeMinimo then
     // Mais uma vez é possível que isso aconteça em função de uma
     // evaporação excessiva em virtude de uma aproximação grosseira no
     // cálculo. Assim é tentada uma segunda vez agora com o
     // o VolFinalInterv = VolumeAtual. Isso é controlado, também, pela
     // variável auxiliar PrimVez.
     if Not PriVez then
        begin
        VolFinalInterv := VolumeAtual;
        Break; // NÃO HÁ ÁGUA - Sai do ciclo
        end
     else
        begin
        VolFinalInterv := VolumeAtual;
        PriVez := False;
        Continue;
        end
  else
    PriVez := True;
  {EXISTE ÁGUA - é chamado o método Opera que contém as regras de
   operação do reservatório. Este método pode ser escrito pelo usuário
   com o propósito de testar regras operacionais.
   Seu funcionamento pode ser combinado com o método Planeja da Classe
   Projeto. É feita a verificação se as demandas planejadas podem ser
   atendidas totalmente ou apenas em parte.
   Como saída este método dá:
     DPO = Demanda Primaria Operada,
     DSO = Demanda Secundária Operada,
     DTO = Demanda Terciária Operada e
     Deflu_OP = defluvio operado (agua que é liberada para jusante)
                IMPORTANTE: Deflu_Op não deve incorporar os RETORNOS das
                Demandas Operadas no Reservatorio - estes são
                considerados como que retornando ao curso d'água à
                jusante do mesmo. Outras variaveis podem ser alteradas
                (ver cabeçalho do método) sob exclusiva responsabilidade
                do usuário.}
VolumeAux := VolumeAtual;
Opera(VolumeAtual, DPP, DSP, DTP, Deflu_Pl, {out} DPO, DSO, DTO,
      Deflu_OP);
  Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
  if DPO > DPT then
     begin
     DPO := DPT;
     // avisa do erro
     end;
  if DSO > DST then
     begin
     DSO := DST;
     // avisa do erro
     end;
  if DTO > DTT then
     begin
     DTO := DTT;
     // avisa do erro
     end;
  // Verifica a CONSISTÊNCIA de saída da OPERA em termos de
  // VIABILIDADE FÍSICA DO VOLUME DO RESERVATÓRIO
  VolumeAtual := VolumeAtual - (DPO + DSO + DTO + Deflu_OP);
  // verifica se o VolumeAtual é inferior ao mínimo
  if VolumeAtual < VolumeMinimo then
     begin
```

```
      // procura atender à limitação do VolumeMinimo reduzindo a
defluência operada
      Deflu_OP := Deflu_OP - (VolumeMinimo - VolumeAtual);
      if Deflu_OP >= 0 then
         // somente a redução da defluência foi suficiente
         // DPO, DSO e DTO continuam atendidas conforme a OPERA
         VolumeAtual := VolumeMinimo
      else
         {NÃO É SUFICIENTE REDUZIR APENAS A DEFLUÊNCIA:
          - esta situação já deveria ter sido tratada em Planeja e/ou
          em Opera através de um critério de racionamento de modo
          que agora não acontecesse esse tipo de inviabilidade.
          Entretanto, esses testes são feitos exatamente para que se
          isso venha a acontecer, não tendo sido tratado
          convenientemente pelo usuário não prejudique o
          desenvolvimento da simulação.}
          begin
         // existe água a ser racionada - é chamado o RACIONA
         // a saída da OPERA é tornada sem efeito
         // o VolumeAtual é reconstituído para o valor anterior à
          utilização da OPERA
          VolumeAtual := VolumeAux;
          AguaDisponivel := VolumeAtual - VolumeMinimo;
          Raciona(AguaDisponivel, DPP, DSP, DTP, {out} DPA, DSA, DTA);
          Projeto.AtualizaPontoExecucao(FNome + '.BalancoHidrico', nil);
            DPO         := DPA;
            DSO         := DSA;
            DTO         := DTA;
            Deflu_OP    := 0 ;
            VolumeAtual := VolumeMinimo;
         end;
      end; // VolumeAtual é menor que VolumeMinimo ...
    // verifica se o VolumeAtual é superior ao máximo
    if VolumeAtual > VolumeMaximo then
       begin
       Vertimento := VolumeAtual - VolumeMaximo;
       VolumeAtual := VolumeMaximo;
       end;
    // verifica CONVERGÊNCIA de VOLUMES para cálculo da EVAPORAÇÃO
    Sair := (ABS(VolFinalInterv - VolumeAtual) <= (0.01 * VolumeMaximo));
    VolFinalInterv := VolumeAtual;
  until Sair or (CicloEvapo > 100);
  // Atualização das Demandas Atendidas.
  FDPA[I]    := Hm3_m3_Intervalo(DPO);
  FDSA[I]    := Hm3_m3_Intervalo(DSO);
  FDTA[I]    := Hm3_m3_Intervalo(DTO);
  FVolume[i] := VolFinalInterv;
  Retorno    := FFRP*DPO + FFRS*DSO + FFRT*DTO;
  FDefluencia[i] := Hm3_m3_Intervalo(Deflu_OP + Retorno + Vertimento);
  FDeflu_OP[i] := Deflu_OP;
  if FGerarEnergia then
     begin
     Queda := CalculaCotaHidraulica((VolumeAtual + VolInicioInterv) / 2)
              - FCotaJusante;
     FEG[i] := CalculaEnergia(Queda, Hm3_m3_Intervalo(Deflu_OP));
     if FEnergiaScript <> nil then
        begin
        FProjeto.AtualizaPontoExecucao(FNome + '.ScriptEnergia',
                                       FEnergiaScript);
        FEnergiaScript.Execute;
        end;
     end;
end;
```

ANEXO 4 - MÉTODOS E FUNÇÕES AUXILIARES CRIADAS PARA A LINGUAGEM PASCAL SCRIPT

Anexo 4.1 FUNÇÃO PCSENTREDOIS.

A função PcsEntreDois foi criada diante da necessidade de criação de uma lista de PCs localizados entre dois outros PCs informados pelo usuário. Essa função foi construída no ambiente Delphi e disponibilizada no Editor Pascal Script do modelo Propagar MOO, sendo apresentada a seguir.

```
function TprProjeto.PCsEntreDois(const NomePC1, NomePC2: String):
 TStrings;
var pc, pc1, pc2: TprPC;
    EncontrouPC2: boolean;
begin
  pc1 := PCPeloNome(NomePC1);
  pc2 := PCPeloNome(NomePC2);
  if pc1 = nil then Raise Exception.CreateFmt(cErro, [NomePC1]);
  if pc2 = nil then Raise Exception.CreateFmt(cErro, [NomePC2]);
  Result := TStringList.Create;
  EncontrouPC2 := False;
  pc := pc1;
  Result.AddObject(pc.Nome, pc);
  while pc <> nil do
    if pc.TrechoDagua <> nil then
       begin
       pc := pc.TrechoDagua.PC_aJusante;
       if pc = pc2 then
          begin
          Result.AddObject(pc.Nome, pc);
          EncontrouPC2 := True;
          Break;
          end
       else
          Result.AddObject(pc.Nome, pc);
       end
    else
       pc := nil;
  if not EncontrouPC2 then
     Result.Clear;
end;
```

Anexo 4.2 MÉTODO REALIZABALANÇOHÍDRICOATE.

Esse método foi criado para que fosse possível realizar um balanço hídrico parcial sobre a rede hidrográfica do modelo. O método realiza o balanço hídrico da rede até um determinado PC informado pelo usuário. Esse método foi construído em ambiente Delphi e é disponibilizado através do Editor Pascal Script do modelo Propagar MOO, sendo apresentado a seguir.

```
procedure TprProjeto.RealizaBalancoHidricoAte(PC: TprPCP);
var i: Integer;
    SubRede: TStrings;
begin
  SubRede := ObtemSubRede(PC);
  for i := 0 to SubRede.Count-1 do
    TprPCP(SubRede.Objects[i]).BalancoHidrico;
  SubRede.Free;
end;


function TprProjeto.ObtemSubRede(PC: TprPCP): TStrings;
var Temp: TList;
  procedure Percorre(PC: TprPCP);
  var i: Integer;
  begin
    for i := 0 to PC.PCs_aMontante-1 do
      Percorre(TprPCP(PC.PC_aMontante[i]));
    Temp.Add(PC);
  end;
var i: Integer;
begin
  Temp := TList.Create;
  Percorre(PC);
  Result := TStringList.Create;
  // Organiza os PCs da Sub-arvore de acordo com os PCs que já
     estao organizados hierarquicamente
  for i := 0 to PCs.PCs-1 do
    if Temp.IndexOf(PCs[i]) > -1 then
       Result.AddObject(PCs[i].Nome, PCs[i]);
  Temp.Free;
end;
```