Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2010

Pedro André dos Santos Gaspar

Laboratório Virtual de Sistemas de Controlo - Realidade Virtual

Pedro André dos Santos Gaspar

Laboratório Virtual de Sistemas de Controlo - Realidade Virtual

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica da Dra. Petia Georgieva , Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof^{a} Doutora Ana Maria Perfeito Tomé
Professora Associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor António José Pessoa de Magalhães
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Prof^{a}. Doutora Petia Georgieva Georgieva
Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos / acknowledgements

A conclusão desta Dissertação de Mestrado em Engenharia Electrónica e Telecomunicações foi possível graças a inúmeras pessoas. Como tal gostaria de agradecer toda a ajuda e motivação que me foi transmitida:

À Dra. Petia Georgieva, Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, por me ter aceitado como aluno de mestrado, por se ter proposto a desenvolver este trabalho comigo, por me ter orientado durante a realização desta dissertação e pela disponibilidade apresentada para me ajudar a dissolver algumas dúvidas sempre que necessitei.

Aos meus pais, pelo suporte contínuo que me deram ao longo desta jornada, pela força e moral que sempre me transmitiram e pelo facto de terem estado sempre presentes.

Ao meu irmão, pelo positivismo e força transmitida durante a realização deste projecto.

A toda a minha família, que sempre mostrou interesse pelo estado em que se encontrava a dissertação e pela motivação com que sempre me apoiaram.

A todos os meus amigos, que sempre acreditaram em mim, me ajudaram sempre que foi necessário e estiveram sempre presentes nos momentos em que necessitei.

As pessoas e profissionais impecáveis das instituições CSA (Clínica de Saúde Atlântico) do Porto e IRF (Instituto de Reabilitação Física) de Ílhavo por me terem recuperado física e psicologicamente, e me terem transmitido força, alegria e auto-confiança.

À Professora e amiga Ana Relvas pela ajuda na revisão da gramática e da sintaxe ao longo desta dissertação.

À minha mãe pelo tempo dispendido na revisão da escrita desta dissertação de mestrado.

Palavras-chave

Laboratório virtual de controlo; Modelos virtuais em VRML; Matlab® - Virtual Reality Toolbox; Modelos virtuais 3D; Compensadores P, PI, PD, PID; Controladores Atraso-Avanço; Compensadores Cancelamento & Implantação de pólos/zeros; Projecto de Servo Sistemas; Controlo de velocidade de um automóvel; Controlo da velocidade angular de um motor DC; Controlo do nível em dois tanques acoplados.

Resumo

Este projecto visa a implementação de um laboratório virtual de sistemas de controlo na aplicação computacional Matlab®, integrando as ferramentas Virtual Reality Toolbox (VRT) e Simulink. Desse modo criaram-se três modelos virtuais de sistemas de controlo e a sua respectiva animação 3D em tempo real:

- Controlo de velocidade de um veículo, cujo controlo foi efectuado usando dois controladores: PI e Atraso.
- O controlo de velocidade angular de um motor DC foi efectuado com um controlador PI, compensador tipo cancelamento & implantação de pólo e avanço.
- Controlo do nível num esquema de dois tanques acoplados, cujo controlo foi realizado através de um P, PI e através de um compensador de servo sistemas com degrau como sinal de referência.

O projecto dos controladores foi executado recorrendo a técnicas de controlo clássico e moderno para compensação de sistemas lineares. Por este facto, primeiro determinaram-se os modelos matemáticos lineares dos três casos de estudo e em seguida fez-se o ajuste dos parâmetros dos controladores baseados nos modelos lineares. Para cada caso, integrou-se a dinâmica com os ambientes virtuais 3D implementados usando as ferramentas de simulação e blocos Simulink® e Virtual Reality Toolbox. Dos resultados experimentais concluiu-se que os modelos implementados demonstram correctamente o funcionamento integrado dos modelos lineares de sistemas de controlo com os modelos virtuais 3D, podendo estes servir os alunos das cadeiras de Sistemas e Controlo I e Sistemas e Controlo II.

Key Words

Virtual Control Laboratory; VRML and 3D virtual models; Matlab® - Virtual Reality Toolbox; P, PI, PD and PID Controllers; Lead-Lag Controllers; Cancel & Placement of poles/zeros; Design of Servo Systems; Car cruise control system; DC motor angular speed control system; Level control in a two coupled tanks system.

Abstract

This project intends to build a virtual laboratory for control systems in Matlab® computational environment, integrating the Virtual Reality Toolbox (VRT) and Simulink®. Thus three control systems virtual models were created, as well as their respective real time 3D animation:

- Car cruise control, whose control was implemented using two types of controllers: PI and Lag controller.
- DC motor angular speed control which was done by a P, a PI, a Cancel & Placement of pole and a Lead controller.
- Level control in a two coupled tanks system, which was controlled by a P and PI controllers and by servo system design.

The controllers were designed through classic and modern techniques for compensation of linear systems. Hence, in first place all three study cases mathematical linear models were determined and then controller parameters were tuned based on these linear systems. For each case, its dynamic was integrated with its respective 3D environment using Simulink® and Virtual Reality Toolbox simulation tools and blocks. The data acquired from the experiments allows to come to the conclusion that the models created show the correct integrated operation of the linear models of the control systems with the 3D virtual scenes created and that this virtual laboratory could serve students of Control Systems I and Control Systems II courses for the next academic year.

Conteúdo

Lista de Figuras

0.1 Lista de Abreviaturas

DC - *Direct Current*

ess - *error in steady state*

FT - Função(ões) de Transferência

NaN - *not a number*

SGV - *Slide Gate Valve*

tr - *rise time*

ts - *settling time*

VL-CSD - *Virtual Laboratories for Control Systems Design*

VRML - *Virtual Reality Modeling Language*

VRT - Virtual Reality Toolbox

VRTS - *Virtual Reality Toolbox Server*

0.2 Prefácio

Laboratório Virtual de Sistemas de Controlo - Realidade Virtual é o título da dissertação que a seguir se apresenta, desenvolvida no âmbito do Mestrado do Curso de Engenharia Electrónica e Telecomunicações, realizado no Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Sendo hoje tacitamente aceite que a electrónica e as telecomunicações são um aliado valioso para que o Homem chegue mais longe no espaço e no tempo, sem perder horas inutilmente e permanecendo em locais muito circunscritos, a atenção recaiu no estudo de uma área que ganha cada vez mais pujança: os laboratórios virtuais, no caso vertente para simulação de sistemas de controlo. Daí que, ao longo de vários meses, a atenção se tenha focado sobre estudos de sistemas de controlo e novas técnicas de ensinamento e aprendizagem de controlo clássico e moderno, procurando-se implementar ambientes virtuais propícios ao teste e simulação de sistemas de controlo virtuais, tal como é proposto pelo título deste trabalho.

Para dar corpo à dissertação, esta foi dividida em cinco capítulos, complementados pelos anexos finais, onde se percorrem as diversas variáveis consideradas para o desenvolvimento do trabalho.

Assim, o primeiro capítulo funciona como uma introdução necessária à compreensão dos sistemas de controlo e o respectivo enquadramento no quotidiano do indivíduo. Paralelamente, definir-se-ão os objectivos da presente dissertação no que diz respeito ao software a utilizar para que se possam atingir os propósitos traçados, bem como focar-se-á a atenção em usos futuros a nível académico e nos conteúdos que se visa abordar com a criação destes laboratórios.

No capítulo seguinte, partindo de uma revisão da literatura e verificando o estado de arte sobre o tema deste trabalho, dá-se conta de um estudo e análise dos laboratórios virtuais de sistemas de controlo já implementados actualmente, porém, necessário já que se aproximam bastante daquele que se pretende aqui criar.

O capítulo terceiro é vital no contexto desta dissertação por ser aqui que se debruça sobre as ferramentas de software disponibilizadas pela aplicação Matlab®, empregues na implementação do laboratório virtual em causa. Aproveita-se para descrever as *toolboxes* utilizadas, bem como os seus blocos funcionais mais relevantes.

No quarto capítulo, debruça-se sobre a descrição dos procedimentos e a implementação de três casos de estudo para o laboratório de sistemas de controlo. Deter-se-à, logo a abrir, no caso relativo ao controlo da velocidade de um veículo, seguido do estudo relativo ao controlo de velocidade angular de saída de um motor DC e, por fim, o terceiro caso de estudo assente no controlo do nível de água num sistema de dois tanques acoplados. Tudo isto, obviamente, considerando o ambiente virtual, que será a base para a concretização do nosso trabalho. Para tal, este capítulo subdividir-se-á, a fim de se traçar também uma abordagem matemática e científica inerente a problemas de controlo, usando a realidade virtual, como referido anteriormente.

Por fim, no último capítulo, apresentam-se as conclusões sobre a evolução sofrida por este projecto, abrindo as portas a futuras sugestões de trabalho, já que não se encerra aqui o estudo que ora se apresentou, graças à multiplicidade de caminhos que é possível percorrer em torno desta problemática dos laboratórios virtuais.

Atendendo à grandeza deste trabalho e aos imensos passos que foi necessário percorrer para se obterem os resultados finais, aproveita-se o capítulo seis para aí apresentar os anexos onde se encontram os algoritmos de controlo que foram necessário executar a fim de se simularem os vários ambientes virtuais traçados por nós e se recolher dados da sua execução.

Em suma, os Laboratórios Virtuais são hoje uma realidade cada vez mais necessária e espera-se que este seja apenas mais um contributo no caminho virtual que a Electrónica e as Telecomunicações há muito criaram.

Capítulo 1

Introdução

1.1 Enquadramento

A industrialização maciça do mundo em que vivemos actualmente tem por base o uso de vastos sistemas de controlo. Este facto, somado aos elevados custos que acarretam o projecto e o teste de mecanismos de controlo, foi o ponto de partida para a criação do conceito de laboratório virtual de sistemas de controlo.

A implementação desta ideia, porém, não seria possível sem a existência de ferramentas de software que permitissem a criação de ambientes simulados. Assim, os avanços a nível tecnológico das aplicações informáticas de simulação e o aparecimento de linguagens de programação capazes de descrever e gerar autênticos mundos virtuais tornaram possível a evolução do conceito de laboratório virtual de sistemas de controlo. Deste modo, a abordagem a projectos da área de controlo de sistemas pode iniciar-se com a implementação e simulação do sistema a nível virtual, sendo possível fazer verificação do funcionamento do mecanismo sem correr riscos de nível monetário e humanitário.

No âmbito particular desta tese de mestrado, o laboratório virtual de sistemas de controlo a implementar visa a criação de modelos de controlo em realidade virtual, tirando partido das funcionalidades do software Matlab® e das suas *toolboxes*. Estes modelos permitirão uma forma inovadora de aprendizagem de técnicas de controlo, mas também uma aproximação do aluno à prática de engenharia de controlo com sistemas "físicos" virtuais.

1.2 Objectivos

Relativamente aos objectivos finais, a presente tese de mestrado tem como principal finalidade a criação de um Laboratório Virtual de Sistemas de Controlo, a partir do software Matlab® e das suas toolboxes, nomeadamente através da Virtual Reality Toolbox (VRT) e do Simulink®.

Os modelos virtuais criados ao longo deste projecto serão desenhados para simulação e teste de algoritmos clássicos de controlo como são os casos de controladores P (Proporcional), PI (Proporcional e Integrador), PID (proporcional, integral e derivativo), compensadores do tipo cancelamento/implantação de pólos/zeros e compensadores atraso-avanço.

Para o último caso de estudo pretende-se ainda efectuar uma pequena introdução ao controlo moderno através do projecto de compensadores para servo sistemas.

O conjunto de modelos construídos constituirá o laboratório virtual que se pretende implementar. Este laboratório terá como finalidade a sua utilização durante a componente prática das disciplinas de Sistemas de Controlo I e Sistemas de Controlo II, ambas ministradas a alunos do curso de Engenharia Electrónica e Telecomunicações.

Capítulo 2

Estado de Arte - Realidade Virtual para Design de Sistemas de Controlo

Após pesquisa feita sobre qual o recente estado científico dos laboratórios virtuais de controlo encontraram-se algumas soluções que implementam este conceito. O campo científico em que se inserem este tipo de aplicações pode estender-se a várias áreas, podendo estas incluir-se em ramos científicos como a química, física, hidráulica, mecânica e tendo como denominador comum a automação e controlo de sistemas [18, 12].

Os exemplos mencionados estão na sua maioria relacionados com o ensino de teoria de controlo a nível académico, mas também com projectos de alunos e investigadores no âmbito do *design* e teste de sistemas de controlo. Algumas excepções são os casos em que o conceito de laboratório virtual de sistemas de controlo está afecto à comercialização de tecnologia direccionada para a aprendizagem de técnicas de controlo.

Em seguida apresentam-se os resultados mais relevantes da pesquisa efectuada.

2.1 VIRTUAL LABORATORIES FOR CONTROL SYSTEMS DESIGN (VL-CSD)

Em [19] podemos encontrar um laboratório virtual de sistemas de controlo. Neste é-nos dado a conhecer uma ferramenta informática, de nome VL-CSD, que pretende dar aos seus utilizadores alvo a possibilidade de aplicarem os conhecimentos teóricos de controlo em ambientes simulados.

Esta ferramenta foi desenvolvida na Universidade australiana de Newcastle pelo Professor Graham Goodwin e pela sua equipa, com base na experiência partilhada por todos estes elementos durante 25 anos. Dado que se trata de uma aplicação frequentemente utilizada por engenheiros de controlo que pretendam continuar a sua formação nessa área, esta foi inserida no mercado pelo ramo comercial da Universidade de Newcastle. Por este motivo[1], a análise deste software não foi realizada eficientemente, sendo no entanto possível perceber o seu funcionamento. Um facto vantajoso evidenciado por este laboratório virtual é a independência da aplicação relativamente a outro software, permitindo aos seus utilizadores bastante flexibilidade no seu manejamento. No entanto, para se utilizarem os vários módulos laboratoriais é necessária a sua compra.

Tendo sido desenvolvida num ambiente académico, a ferramenta VL-CSD, tenta implementar uma filosofia de aprendizagem inovadora, na qual são geradas situações experimentais realistas e bastante desafiadoras.

Os casos implementados pela ferramenta VL-CSD foram divididos em dois tipos de abordagens [11]: a primeira engloba exemplos à escala industrial, ou seja, todas as tarefas são realizadas virtualmente através da aplicação, a segunda abordagem, disponibiliza sistemas físicos que proporcionam a realização da experiência e tem-se como principal objectivo aplicar ou demonstrar alguns conceitos específicos de teoria de controlo.

[1] Foi enviado um pedido de uma versão experimental por correio electrónico para a Universidade de Newcastle (Austrália), porém isso não foi possível dado que o software se encontra em revisão e em fase de pré-comercialização.

Como é referido em [11], no desenvolvimento da aplicação teve-se em consideração alguns aspectos característicos em sistemas de controlo reais, nomeadamente, a saturação, as não linearidades, o ruído e as zonas de funcionamento mortas.

Entre os exemplos fornecidos podemos encontrar um sistema de corte de papel, a dinâmica associada a um *rocket* [16], um sistema de redução de espessura e alteração de propriedades de tiras metálicas [18], um sistema quantificador de ruído, um servomecanismo electromecânico [13], um fundidor de metais [12] ou ainda um pacote de controlo da dinâmica do nível de um fluído [15]. As imagens abaixo, retiradas do site já mencionado, demonstram a interface da aplicação e a montagem experimental virtual de alguns exemplos disponíveis.

Virtual Industrial Laboratories

2.1.1 Continuous Caster Laboratory – Classical Control Design

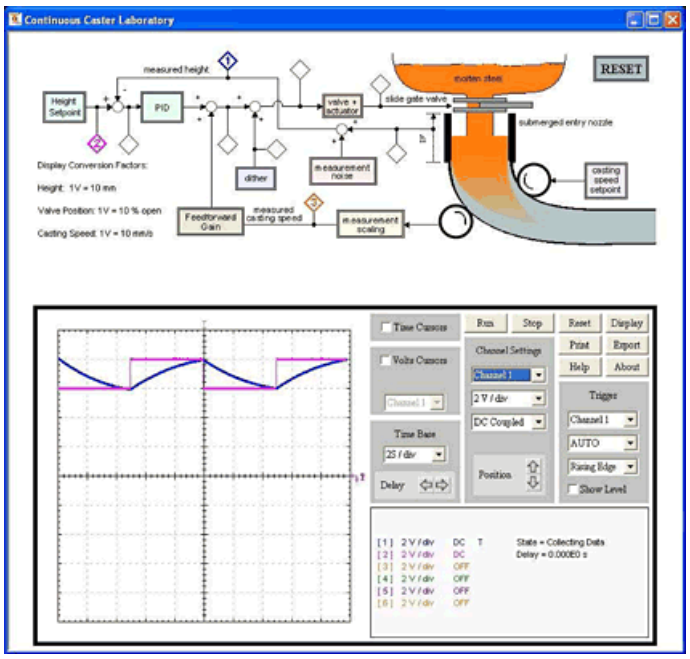

Figura 2.1: Dinâmica de um sistema de fundição contínua e consola de controlo (retirado de [12])

O laboratório virtual [12] (ver figura 2.1) tem como objectivo a construção de barras metálicas rectangulares com uma determinada secção transversal a partir de um molde. Neste exemplo é apresentado o modelo conceptual do sistema através de um diagrama de blocos. Dos blocos que constituem o sistema destaca-se um controlador PID que controla o actuador na válvula SGV (*Slide Gate Valve*)[2]. Abaixo da descrição simbólica da dinâmica do sistema, é apresentada uma consola para escolha dos sinais que se pretendem visualizar, assim como o canal do osciloscópio virtual no qual se desejam obter. A partir deste painel é ainda possível ajustar as escalas de cada canal, a base de tempo de funcionamento do sistema e seleccionar o canal por onde se pretende efectuar o *trigger* (sincronização). Com estas funcionalidades podemos obter várias características interessantes para compreensão de sistemas de controlo tais como, tempos de estabelecimento, valores de pico da resposta, erro relativamente à altura pré-definida e a sobre-elevação do sinal de saída.

O sistema industrial que implementa a fundição de metal, parte da condição inicial em que a base do molde está fechada. Este molde é preenchido com metal fundido a um ritmo constante enquanto sincronizadamente o metal derretido é arrefecido por jactos de água. Quando se inicia o processo de solidificação, a base do molde é reaberta de modo a ser extraída lentamente a barra de metal

[2] Usando uma SGV (Slide Gate Valve) é controlado o ritmo com que é preenchido o molde, pois a abertura da válvula é variável.

num estado semi-sólido. Este passo leva a que no cimo do molde se liberte espaço, o qual vai ser preenchido novamente com o fluído metálico. Como está patente, todo o sistema requer a execução sequencial e sincronizada de processos, os quais falhando o seu propósito podem levar à destruição da própria máquina colocando igualmente em perigo vidas humanas. Por este facto é imperativo que se efectue um controlo preciso do sistema. O exemplo fornecido assume que todo o processo de arrefecimento, bem como a extracção da parte semi-solidificada da barra metálica a partir do molde está afinado e funciona correctamente. Assim, é apenas necessário controlar a *Slide Gate Valve* (SGV) (ver figura 2.2), responsável pelo controlo da quantidade de aço fundido que é depositado no molde. Sendo assim, o principal objectivo será garantir que não é colocada uma quantidade de aço fundido superior ao espaço disponível no molde, mas também que o nível de metal no molde não varie entre limites demasiados distantes, pois este facto pode levar à contaminação do metal, por exemplo, por oxidação.

O funcionamento geral da máquina fundidora assenta em técnicas de controlo clássico, recorrendo essencialmente a malhas de realimentação, projecto de controladores PID ou projecto de funções de transferência (FT).

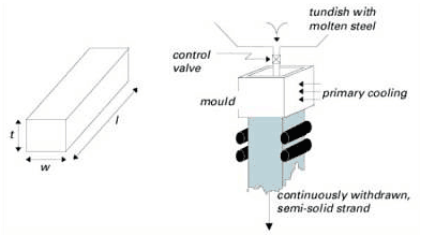

Figura 2.2: Esquema de funcionamento da válvula SGV (*Slide Gate Valve*). (Retirado de [12])

2.1.2 Rolling Mill - System Modeling and Classical Control Virtual Laboratory

Outro exemplo fornecido no pacote de laboratórios virtuais industriais é uma máquina laminadora de metal, cujo principal objectivo é diminuir a espessura e alterar algumas propriedades metalúrgicas das folhas metálicas (figura 2.3).

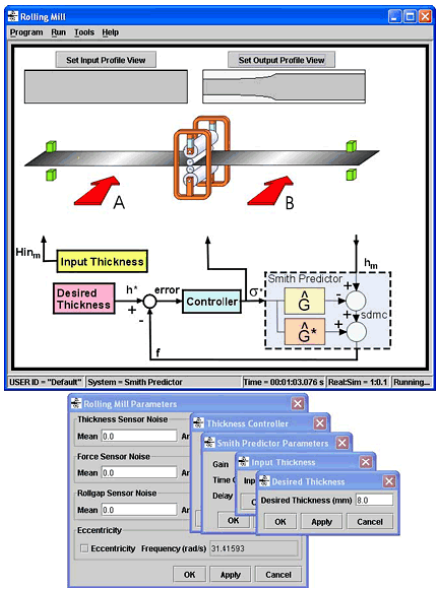

Figura 2.3: *Rolling Mill*. (Retirado de [18])

No caso apresentado na figura 2.3 denotam-se algumas diferenças para o exemplo comentado no ponto 2.1.1. Neste segundo exemplo é também apresentado um esquema geral da montagem experimental e ainda um diagrama de blocos que representa a dinâmica do sistema. Adicionalmente são abertas cinco caixas de diálogo, as quais permitem ao utilizador não só especificar valores para a espessura desejada e afinar os parâmetros do controlador de espessura, mas também inserir ruído e algumas não linearidades que irão afectar toda a dinâmica do sistema.

2.1.3 Rockets - Controller Design Virtual Laboratory

Para implementação do controlo de voo de um *rocket* podemos observar o exemplo em [17]. O rocket virtual foi concebido de duas formas distintas, permitindo ao utilizador proceder ao afinamento do controlador estabilizador de voo para um modelo linear ou para um modelo que assume perturbações, ou seja, um modelo real. Neste exemplo específico espera-se que o aluno já tenha tido contacto com o laboratório virtual [16], no qual é estudada toda a dinâmica de voo de um sistema de engenharia aeroespacial como são os *rockets*.

Com este laboratório virtual (figura 2.4) os autores pretendem que os utilizadores da aplicação relacionem a dependência que existe entre o projecto de parâmetros e a estabilidade do voo de um *rocket*. Durante este estudo é esperado que o aluno desenvolva capacidade de linearização de modelos baseados em equações dinâmicas não lineares.

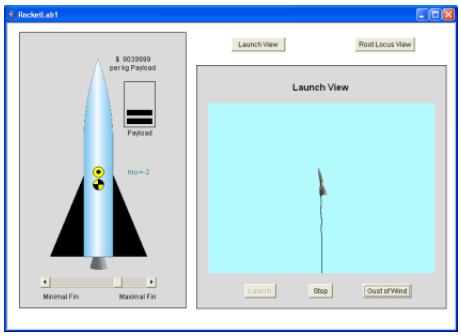

Figura 2.4: *Rockets – Dynamics Virtual Laboratory.* (Retirado de [16])

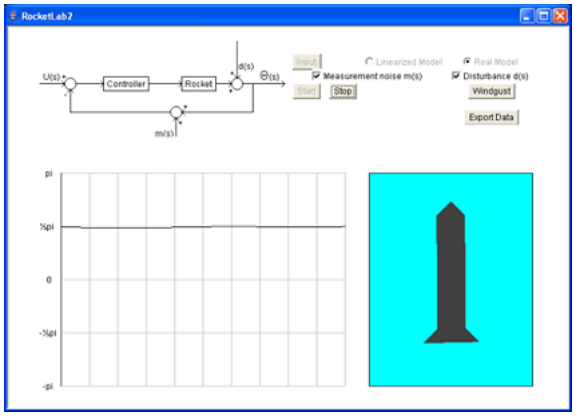

Figura 2.5: Laboratório para design do controlador estabilizador de voo de um *rocket.*(Retirado de [17])

Por outro lado, na interacção com o modelo em [17](ver figura 2.5) é pretendido que se perceba a instabilidade do sistema em malha aberta e se arquitecte uma melhor performance do voo do *rocket* através do *design* de um controlador para um sistema em malha fechada.

Virtual Benchtop Laboratories

2.1.4 Electromechanical Servomechanism - Classical Design Virtual Laboratory

Os autores do software VL-CSD (Virtual Laboratories for Control System Design) criaram ainda um segundo tipo de módulos de laboratórios virtuais. Estes são experiências mais simples em que o utilizador interage directamente com um sistema físico, usando a interface criada para realizar o controlo do sistema. O servomecanismo electromecânico é um exemplo que respeita estas características, tendo sido criados três modelos para aplicação de diferentes técnicas de controlo [13](ver figura 2.6). Entre os modelos disponibilizados podemos encontrar um que faz uso das regras de Ziegler-Nichols[3] [23, 14] para afinação de parâmetros do controlador.

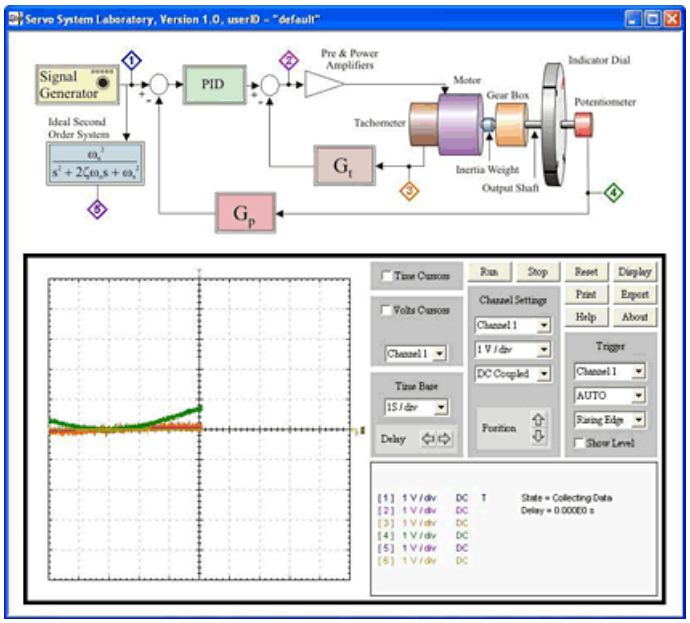

Figura 2.6: Laboratório virtual de *design* clássico de um servo electromecânico. (Retirado de [14])

No caso específico do modelo clássico do servomecanismo, o objectivo é estudar a dinâmica relacionada com o funcionamento de um servomecanismo electromecânico cujo núcleo é um motor DC (Direct Current)[4]. Para esse fim foram acopladas a este modelo algumas não linearidades: *wrap-around*[5] do potenciómetro, sinal de ruído e saturação na saída do amplificador de potência. Para visualização e teste desta experiência, é utilizado um kit (Servo Kit: ver figura 2.7) que permite fazer o controlo de posição através da execução da aplicação. O módulo da ferramenta VL-CSD dedicado a este exemplo permite a realização de estimativas dos valores para os parâmetros K_m, K_p, K_t e τ de modo a controlar o ângulo do eixo do motor DC (*Direct Current*).

[3] Método de afinação de parâmetros de controladores PID, no qual os ganhos (proporcional, integrativo e derivativo) dependem das medições realizadas com malha de realimentação unitária, do ganho crítico, Kc, e do período de oscilação crítico Pc.

[4] Os motores DC (direct current) são utilizados frequentemente em caixas de velocidades de automóveis, actuadores de válvulas e aplicações de robótica.

[5] Extensão do ângulo limite varrido pelo potenciómetro. Ou seja, desintegração dos seus limites de rotação; passagem automática do limite superior do potenciómetro para o limite inferior ou vice-versa.

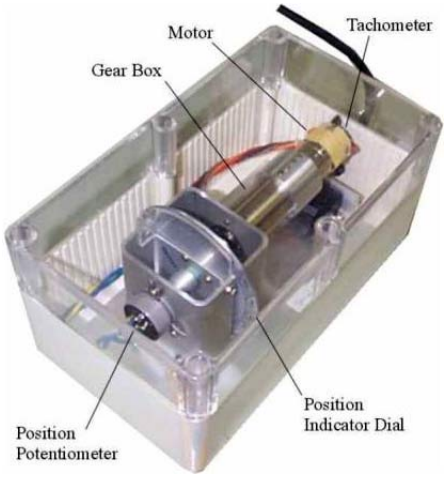

Figura 2.7: Servo Kit. (Retirado de [14])

Este tipo de aplicação torna-se útil em vários esquemas de controlo, nomeadamente, em sistemas robóticos. Um caso concreto seria o acoplamento de uma câmara ou um determinado sensor detector de IR (*Infra-Red*) ao potenciómetro para, no primeiro caso, controlar o ângulo de visão de um mecanismo de vigilância e no segundo, por exemplo, fazer a detecção do farol no concurso Micro-Rato[22].

2.1.5 Fluid Level Dynamics and Control - Coupled Tanks Control Virtual Laboratory

O simulador criado para o laboratório de dois tanques acoplados pretende que seja estudado, tal como o nome indica, um sistema de controlo da dinâmica de um fluído. Estes sistemas são frequentemente usados como *buffers* de entrada de outro módulo que lhes seja subsequente de modo a uniformizar o curso de todo o processo. Noutros casos, o acoplamento de tanques é usado em indústrias de armazenamento de fluídos[15].

No ambiente simulado como o descrito anteriormente, o objectivo prende-se com o controlo das válvulas e bombas associadas a cada tanque. Cada tanque tem uma bomba, a qual por sua vez é controlada por um motor DC. As válvulas permitem o uso de cada tanque individualmente. Na figura 2.8 apresenta-se um esquema físico do sistema implementado.

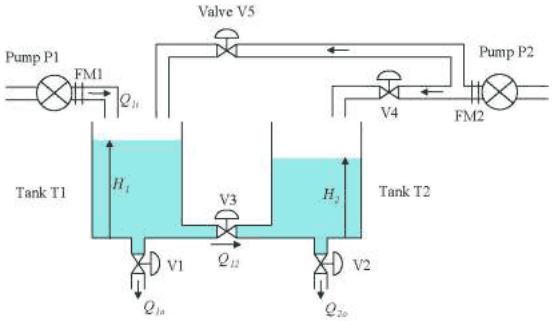

Figura 2.8: Esquema físico do sistema de acoplamento de 2 tanques. Retirado de [15]

O correcto funcionamento deste sistema depende predominantemente da afinação de um controlador PID que actua directamente sobre o motor DC e consequentemente, na bomba P1. A afinação correcta do controlador limita o fluxo de fluído para o tanque T1 dependendo da altura de água H1 medida[6]. A realimentação negativa do valor da altura da água permite ter à entrada do controlador PID um sinal de erro, o qual fornece dados da resposta do sistema relativamente aos sinais de referência.

[6] Os valores de altura de água H1 e H2 são medidos através de um sensor piezoeléctrico de profundidade.

2.2 THE ECOSSE CONTROL HYPERCOURSE - THE VIRTUAL CONTROL LABORATORY

O laboratório virtual encontrado em [6] expõe um conceito de laboratório virtual para sistemas de controlo ligeiramente diferente dos casos descritos anteriormente. Este modelo baseia-se numa abordagem mais simples no que diz respeito à visualização e controlo do sistema físico. Assim, apesar de em alguns casos nos permitir modificar e afinar os controladores usados, apenas se apresenta uma resposta em termos gráficos.

Tal como no exemplo do VL-CSD (Virtual Laboratories for Control System Design) da Universidade de Newcastle[19], uma disciplina de sistemas e controlo da Universidade de Edimburgo fornece vários exemplos para estudo de técnicas de controlo. O caso específico publicado em [6] tira partido da acessibilidade à web. Ou seja, neste caso não há necessidade de adquirir as várias experiências e o software que pode estar associado às mesmas, sendo apenas necessário visitar o site da disciplina ministrada nesta universidade. Outro factor relevante prende-se com o facto de este sítio na internet estar disponível para qualquer utilizador e não apenas para os alunos daquela universidade.

2.2.1 Hot Water Tank Experiment

A experiência presente em [7] tenta recriar um depósito de água aquecida passível de ser usado numa habitação. O esquema geral do sistema a controlar é apresentado na figura 2.9.

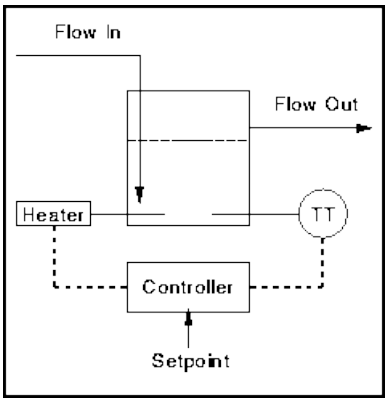

Figura 2.9: Esquema da experiência de um tanque de água quente. (Retirado de [6])

Nesta experiência pretende-se controlar a temperatura da água que sai do tanque de acordo com um determinado *setpoint*[7] . O ponto de funcionamento varia consoante a temperatura a que se pretende a água, com a quantidade de água que se está a requisitar ao tanque mas também com o desvio que se definiu em relação à temperatura. A técnica usada baseia-se num controlo *on-off* à potência fornecida ao aquecedor. De acordo com os parâmetros definidos obtém-se um gráfico que demonstra as medições no sistema (ver figura 2.10).

Executada a experiência são fornecidas explicações teóricas, bem como conclusões relativamente ao método de controlo usado. Deste exemplo conclui-se que a técnica de controlo *on-off* permite obter uma performance correcta do sistema, visto que não necessitamos de elevada precisão na temperatura da água. No entanto, verifica-se que este método de controlo provoca sempre oscilações no sinal de saída, o que para outros tipos de mecanismos pode ser intolerável, e obriga ainda a um compromisso entre a precisão pretendida e o desgaste da válvula do mecanismo de aquecimento.

[7] Setpoint pode ser definido como o valor do sinal de referência. Neste caso concreto poderia ser a temperatura a que se deseja a água, a quantidade de água pretendida e/ou a variação máxima relativamente a esses valores.

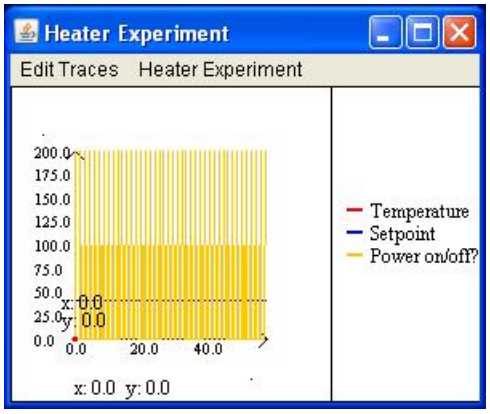

Figura 2.10: Controlo *on-off* do aquecedor e temperatura da água debitada pelo tanque. (Retirado de [6])

2.2.2 Level Control Experiment

A actividade implementada neste segundo caso [8] traduz-se por um sistema de controlo de nível de um fluído, cujo esquema pode ser visto na figura 2.11.

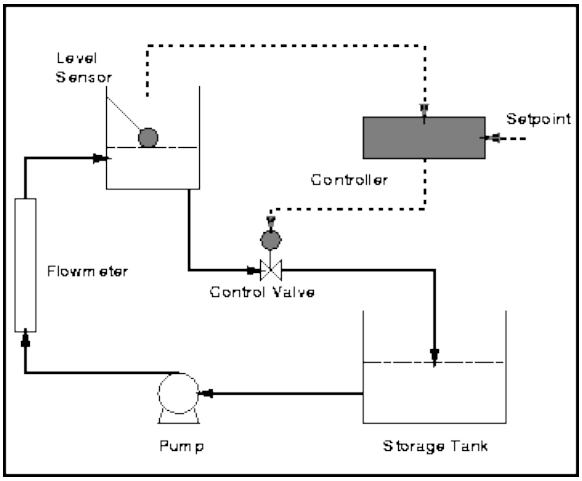

Figura 2.11: Esquema do sistema controlador do nível de fluído num tanque. (Retirado de [6])

No sistema acima o objectivo é actuar utilizando várias técnicas de controlo. Para tal, o laboratório virtual criado faz uma abordagem faseada ao problema. Inicialmente, é proposta a utilização do controlo tipo *on-off*, no qual o utilizador tem a possibilidade de definir o fluxo de entrada no tanque, o nível de referência e a largura de banda (*deadzone*[8]) do controlador que actua na válvula. Seguidamente, é possível fazer o controlo do sistema através de um controlador tipo P ou PI, actuando na interface apresentada pela figura 2.12.

[8] O conceito de *deadzone* pode ser definido como o tempo em que a válvula está aberta, ou seja, faz o escoamento do fluído para o tanque auxiliar de armazenamento. De facto, quanto mais demorado for a *deadzone* maior a largura de banda e maior a variação relativamente ao *setpoint*.

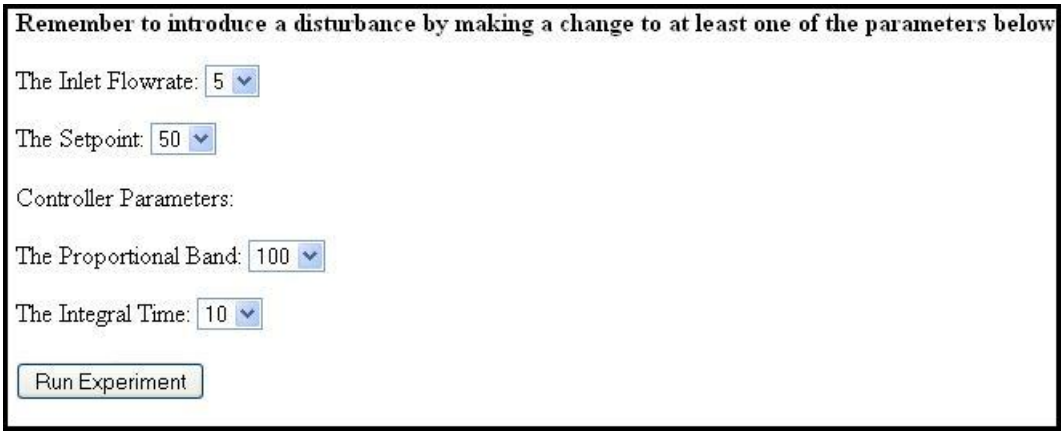

Figura 2.12: Interface de utilizador para afinação dos parâmetros do controlador PI. (Retirado de [6])

Depois de alterados os parâmetros, as condições iniciais e as constantes do controlador, é necessário executar a experiência para que se possam obter resultados. Esses resultados são, mais uma vez, apresentados sob a forma de um gráfico como ilustra a figura 2.13.

Como ficou patente, o conceito de laboratório virtual de sistemas de controlo explorado em [6] permite uma análise de uma vasta gama de técnicas de controlo e apresenta ainda uma abordagem teórica que contempla perguntas e respostas muito pertinentes. Este último ponto é uma mais-valia neste tipo de laboratórios pois serve de suporte à percepção do funcionamento de determinados algoritmos de controlo.

Por outro lado, o facto de não ser visualizado o comportamento físico do sistema, assinala-se como uma desvantagem do laboratório virtual descrito nas linhas acima.

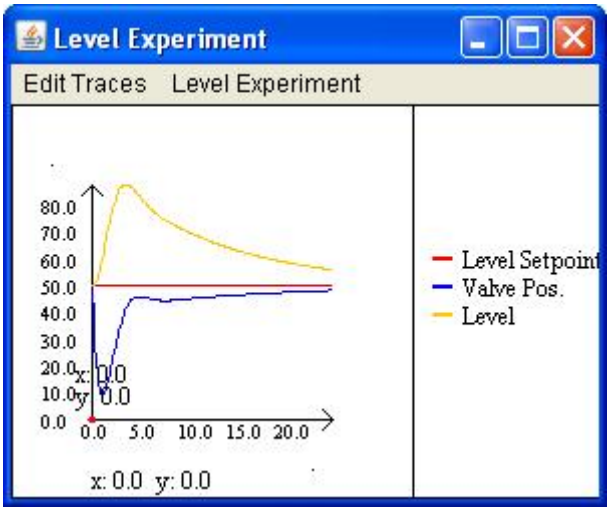

Figura 2.13: Variação da posição da válvula e do nível de fluído para o nível de referência definido. (Retirado de [6])

2.3 VIRTUAL CONTROL LAB 3.1 (VCLAB 3.1)

Um outro laboratório virtual de sistemas de controlo é apresentado em [32]. Este exemplo encontra-se numa plataforma de ensino de conteúdos para engenharia de controlo da Universidade de Bochum da Alemanha. Este efectua uma aproximação aos sistemas de controlo reais através de uma implementação virtual dos mesmos. A implementação destes modelos 3D realizou-se utilizando a VRML (*Virtual*

Reality Modeling Language)[9].

As funcionalidades apresentadas por este laboratório virtual assemelham-se aquelas que se pretendem implementar para a elaboração desta tese, pois faz uso do software Matlab® e das suas *toolboxes*.

Este laboratório apresenta, para além de uma base teórica sólida, vários exemplos de sistemas de controlo, nomeadamente um modelo de três tanques acoplados (figura 2.14), um sistema bola-haste (ver figura 2.15) e um pêndulo com giroscópio (ver figura 2.17).

2.3.1 Laboratory Plant of Three Tanks

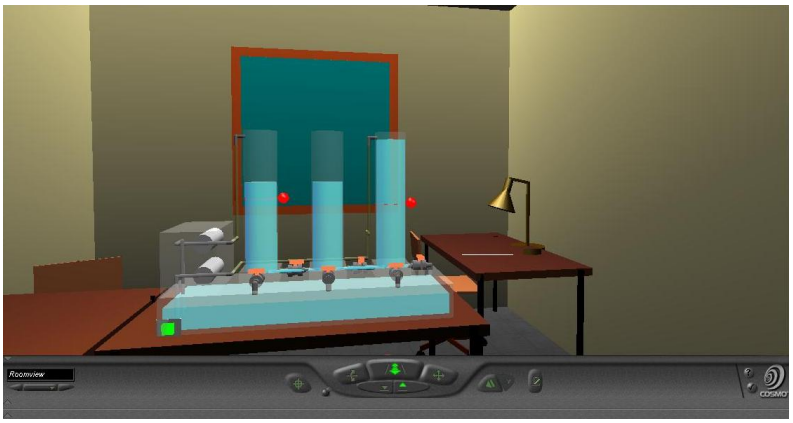

Figura 2.14: Modelo virtual de um sistema de três tanques acoplados. (Retirado de [32])

Similarmente ao modelo citado na página 16 , este caso de estudo tenta exemplificar o controlo de nível de um fluído. O caso específico apresentado caracteriza-se pelo facto de ser necessário controlar o nível (L1 e L2) em dois tanques simultaneamente, fazendo uso de um tanque auxiliar T2, de um depósito situado na base dos três tanques, duas bombas (P1 e P2) ligadas independentemente aos tanques T1 e T3 e um conjunto de seis válvulas que interligam os três tanques entre si e cada um deles ao depósito. O controlo é efectuado a partir dos valores dos sensores de nível (representados na figura 2.14 pelas duas esferas no tanque T1 e T3) e pelas bombas (P1 e P2), que "gerem" a água transferida a partir do depósito para o tanque T1, para o tanque T3 ou para ambos.

2.3.2 Laboratory Plant of Ball and Beam

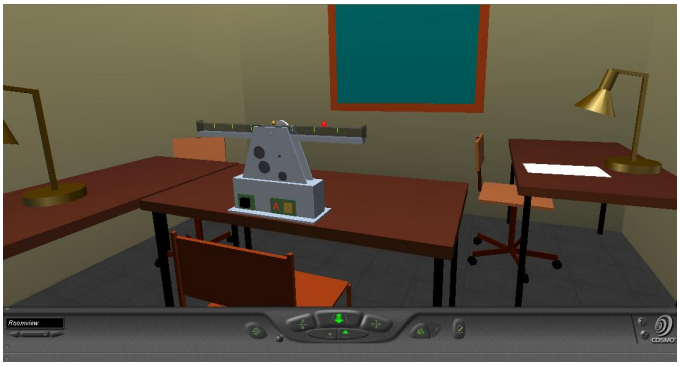

Figura 2.15: Modelo virtual do mecanismo bola-haste. (Retirado de [32])

[9]VRML ou *Virtual Reality Modeling Language* é uma linguagem de programação em modo de texto usada para descrever objectos e as suas características: cor, textura, brilho, transparência, etc. Com a linguagem VRML é também possível munir os objectos de sensores podendo assim mudar o seu estado dinâmico[9].

Na experiência virtual com o mecanismo bola-haste pretende-se equilibrar a bola na haste tendo em conta a posição do peso (representado pela esfera na figura 2.15). O funcionamento do sistema baseia-se num conjunto de engrenagens controladas por um motor DC. Este actua sobre as engrenagens de modo a controlar a posição da haste e mantendo em equilíbrio a esfera que rola na sua superfície. Para diferentes posições do pequeno peso vamos ter um ângulo α variável, relativamente ao plano horizontal em que o sistema está apoiado. Este ângulo vai provocar o deslocamento da bola sobre a haste.

Um esquema simplificado da dinâmica deste sistema mas com um funcionamento diferente pode ser observado na figura 2.16.

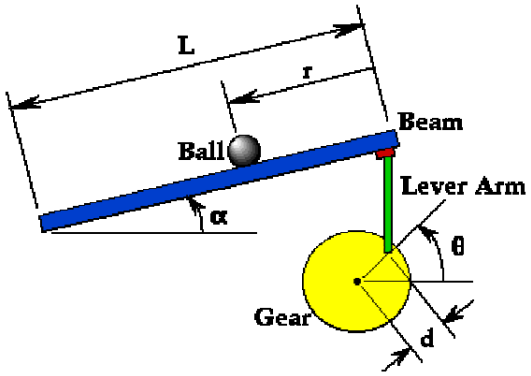

Figura 2.16: Esquema simplificado do sistema bola-haste (retirado de IPT [21])

Considerando desprezável o atrito e o deslizamento da bola na haste, podemos descrever matematicamente o movimento da bola pela seguinte equação:

$$\frac{J}{R^2}\ddot{r} + \ddot{r}m + mg\sin\alpha - mr(\dot{\alpha})^2 = 0 \tag{2.1}$$

A equação 2.1, retirada de [21], rege o movimento da bola na haste tendo em conta a variação do ângulo α aquando da elevação da haste para uma determinada posição, e os valores de constantes como o momento de inércia da bola (J) e o seu raio R . Pela Lei de Newton, pretende-se que o somatório das forças aplicadas a mantenham em equilíbrio, ou seja, a aplicação de controlo na haste tem que ser tal que a esfera fica com aceleração nula.

Por outras palavras, o intuito da experiência é actuar no motor DC que controla o movimento das engrenagens de forma a equilibrar a esfera na figura 2.16.

2.3.3 Laboratory Plant of Gyro Pendulum

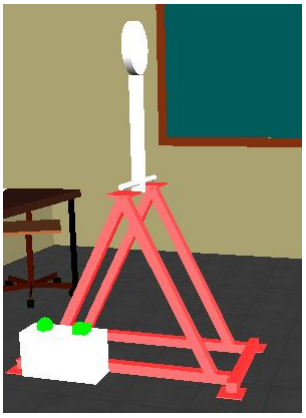

Figura 2.17: Modelo virtual do pêndulo com giroscópio. (Retirado de [32])

Na experiência virtual do giroscópio com pêndulo, visível na figura 2.17, o objectivo é colocar o braço do pêndulo perpendicular ao seu eixo de rotação. Neste caso, o facto de ser usado um giroscópio que roda sobre um eixo colinear com o braço do pêndulo, vai provocar uma variação na posição de equilíbrio do pêndulo. Neste exemplo há ainda a possibilidade de alterar as condições iniciais, pois é permitido mudar a posição inicial do giroscópio, assim como o ângulo inicial da barra do pêndulo.

As técnicas de controlo utilizadas nesta aplicação assemelham-se às usadas no caso do pêndulo invertido. Na prática, este exemplo enquadra-se na tecnologia de funcionamento do já sobejamente conhecido Segway PT[4]. Efectivamente, o Segway Personal Transporter (SPT) recorre a cinco microgiroscópios, funcionando estes como: sensores de movimento, sensores de inclinação e motores eléctricos controlados por dois microprocessadores que executam software específico, controlando assim o veículo e equilibrando o seu utilizador.

Da análise efectuada a este género de laboratório virtual de sistemas de controlo, conclui-se que esta seria uma abordagem vantajosa, visto que obtemos uma maior proximidade entre o conhecimento da teoria de controlo e a aplicação dos mesmo na prática. Por outro lado, existem algumas desvantagens de implementação, sobretudo pelo uso de uma versão já ultrapassada do software Matlab® (versão máxima recomendada Matlab 5.x); a utilização de uma versão mínima 2.x da toolbox Simulink® do Matlab®; e o uso do *browser* Netscape Navigator® (versão máxima aconselhada Netscape 4.x) com recurso a um *plug-in*[10] para o processamento de ambientes virtuais. A utilização obrigatória destas ferramentas tornam esta plataforma um pouco obsoleta.

[10] Para o Virtual Control Lab 3.1 são aconselhados dois plugins disponíveis para o browser Netscape Navigator®: o SGI Cosmo Player 2.x ou o Cortona 3D Viewer.

Capítulo 3

Descrição das Ferramentas de Trabalho

3.1 Introdução

Neste ponto serão analisadas todas as ferramentas de trabalho usadas ao longo da implementação do laboratório virtual de sistemas de controlo.

A ferramenta Matlab® (MATrix LABoratory) foi inicialmente criada no fim dos anos 70 por Cleve Moler na Universidade do Novo México, aquando da sua presidência do Departamento de Ciências da Computação, para engenheiros ligados ao ramo de projecto de controladores. Dado o seu rápido alargamento a outros domínios, o objectivo deste software passou a ser também facilitar as exigências do cálculo numérico, ou seja, permitir facilmente o manuseamento de matrizes, o processamento de sinais e construção de gráficos através de uma descrição directa do problema matemático.[29]

Mais tarde, e devido ao crescente uso desta ferramenta, o programa Matlab® foi reescrito em linguagem C pelos fundadores da Mathworks (Jack Little, Cleve Moler e Steve Bangert).

O Matlab® usa uma linguagem de programação com o seu nome (ou vulgarmente denominada de código-M), a qual conjuga funções matemáticas gráficas e uma linguagem de alto nível. Esta linguagem de programação traduz abordagens matemáticas a grandes quantidades de dados em resultados gráficos. O resultado é uma ferramenta poderosa usada actualmente em áreas como projecto de sistemas de controlo, processamento de sinais, processamento de imagem e análise de modelos financeiros.[26]

As características que tornam a programação em Matlab® tão simples são o facto de não ser necessário fazer uma pré-declaração ou inicialização das variáveis ou do seu tipo, a forma de expressão se assemelhar às expressões matemáticas escritas e ainda oferecer uma vasta gama de funções que permitem tratar dados numéricos de diferentes formas.[29, 5]

Actualmente, esta linguagem de computação técnica apresenta um ambiente gráfico interactivo que combina múltiplas funções de modelação e projecto de sistemas através das *toolboxes* fornecidas. Para modelação e simulação de modelos esta ferramenta traz a aplicação Simulink®.

No âmbito desta tese são usadas especificamente as ferramentas Simulink® e VRT (Virtual Reality Toolbox), incluídas na versão 7.5.0.342 (R2007b) do software Matlab®. Por esta ordem vão-se descrever as ferramentas Simulink® (apenas os seus blocos mais relevantes) e VRT. Relativamente a esta última vão analisar-se os blocos funcionais e o seu funcionamento interno, o qual, por depender directamente da linguagem de progamação VRML, apresenta alguns padrões a ter em consideração quando se pretender integrar os mundos virtuais criados com as funcionalidades do Matlab®.

3.2 Simulink®

3.2.1 Introdução à *Toolbox* Simulink®

O principal objectivo da consola Simulink® do Matlab® é permitir a modelação, análise e simulação de sistemas dinâmicos.

A ferramenta Simulink® permite a criação de um modelo de um sistema físico a partir de um diagrama de blocos. A sua interface permite o acoplamento de diferentes tipos de blocos através do rato, bem como a alteração dos parâmetros dos blocos através do teclado. A configuração dos parâmetros pode ser feita, alternativamente, através de um *script* que deve ser executado previamente em ambiente Matlab® ou, simplesmente, pela execução das expressões na janela de comandos.

Para a realização do diagrama de blocos do sistema pretendido é necessário que este tenha sido modelado matematicamente. A ferramenta Simulink® inclui vários blocos de diferentes tipos de sistemas, os quais requerem apenas a definição de algumas constantes e tratamento dos sinais de entrada e saída . As áreas de destaque presentes nesta aplicação são: exploração aeroespacial, aplicações militares de defesa, indústria automóvel, sistemas de telecomunicações e electrónica, processamento de sinal e instrumentação médica. Para além de todas estas opções, a interface gráfica permite a criação de novos blocos. Numa outra perspectiva, a visualização global do sistema pode ser feita seguindo uma abordagem *top-down* ou *bottom-up*, permitindo deste modo, ter uma melhor noção de como está organizado o sistema e quais os blocos funcionais que interagem.[27]

Outra particularidade do suplemento Simulink® é a sua capacidade de resolução de problemas a nível de sistemas não lineares, expandindo assim ainda mais a aplicabilidade desta ferramenta.

Dada a versão do Matlab® utilizada na construção do laboratório virtual de sistemas de controlo em realidade virtual, foi usada a versão 7.0 (R2007b) do Simulink® para a realização dos modelos dos sistemas a serem implementados em realidade virtual.

3.2.2 Blocos Fundamentais do Simulink®

Para a construcção dos modelos físicos que definem a dinâmica de um determinado sistema de controlo, a ferramenta Simulink® permite a interacção entre blocos de diferentes *toolboxes*.

Neste subcapítulo pretende-se dar a conhecer alguns dos principais blocos funcionais da ferramenta Simulink® que permitem o desenvolvimento e simulação de sistemas de controlo reais. Nesse sentido apresentam-se em seguida os blocos mais relevantes e uma descrição do seu funcionamento.

A figura 3.1 mostra os blocos Simulink® de uso mais frequente. Dentro deste conjunto encontram-se blocos extremamente úteis para modelação do diagrama de blocos do sistema que se pretende implementar. Tendo em conta a sua utilidade, os casos mais concretos são a utilização do bloco "Scope" que permite visualizar a evolução dos sinais ao longo do tempo e fazer assim o *debugging* do comportamento do sistema; o bloco "Gain", responsável por efectuar o produto do sinal por um valor de ganho constante ou calculado "*online*", de acordo com variáveis do sistema, podendo este ser definido sob a forma de escalar ou na forma matricial; os blocos "In1" ou "Out1" capazes de tornar acessíveis qualquer sinal que circule no sistema, permitindo ainda transformá-los em variáveis de entrada ou saída de um dado subsistema; o bloco "Subsystem" que permite agregar num único bloco um conjunto de outros blocos e deste modo facilitar a compreensão do funcionamento geral de um dado sistema; os blocos "Product" e "Sum" que facilitam a realização de cálculos matemáticos como produtos, somas e subtracções; finalmente os blocos "Mux" e "Demux", os quais permitem respectivamente, fazer a multiplexação e desmultiplexação de sinais, ou seja, no primeiro caso agrupar diferentes sinais num só (sinal matricial) e num segundo caso realizar a operação inversa, isto é, dividir um sinal único em vários sinais vectoriais.

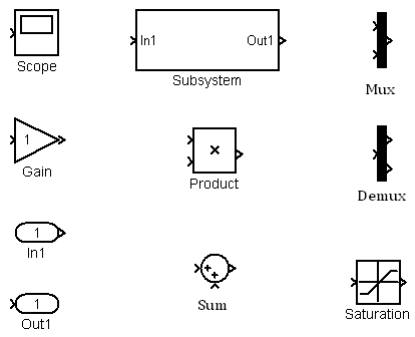

Figura 3.1: Blocos frequentemente utilizados

Como é visível na imagem 3.2, a toolbox Simulink® disponibiliza ainda outro tipo de blocos de carácter geral. Estes realizam outro tipo de tarefas quando acoplados a outros blocos.

Situados mais à esquerda encontram-se blocos cuja principal função será fornecer dados/sinais ao sistema. Estes sinais vão ser depois tratados através de operações matemáticas por forma a conduzirem correctamente o comportamento do sistema. A este nível a ferramenta Simulink® apresenta uma vasta gama de soluções, as quais permitem desde a utilização de sinais de teste como constantes (bloco "Constant"), degraus (bloco "Step") ou rampas (bloco "Ramp") ao uso de variáveis que se encontrem no directório de trabalho corrente (bloco "From Workspace" ou bloco "From File").

Na segunda coluna de blocos estão apresentadas algumas operações matemáticas que se adequam às necessidades não cobertas pela biblioteca de blocos apresentada na figura 3.1.

O projecto de sistemas de controlo a nível digital recorre, usualmente, a funções de transferência (FT) que descrevem o funcionamento do sistema através da aplicação da Transformada de Laplace às equações que regem a sua dinâmica. A utilização das funções de transferência no domínio de Laplace é outra particularidade da ferramenta Simulink®. Esta funcionalidade está directamente relacionada com os blocos apresentados na terceira coluna do conjunto apresentado anteriormente. A sua utilização implica apenas que o utilizador conheça as leis que regem a dinâmica do sistema e qual a variável do sistema que pretende controlar. Depois de obtida a FT no domínio de Laplace e em ordem à variável que se deseja controlar, todo o processo de utilização do bloco "Transfer Fcn" é trivial, sendo somente necessário preencher os coeficientes dos polinómios em "s" do numerador e do denominador da FT calculada.

Adicionalmente, podem ser utilizados os blocos situados mais à direita. Os dois primeiros desempenham as funções inversas dos blocos "From Workspace" e "From File". Ou seja, a partir do sistema que estamos a analisar podemos colocar em memória os sinais que pretendermos, quer através da gravação numa variável do espaço de trabalho quer para um determinado ficheiro.

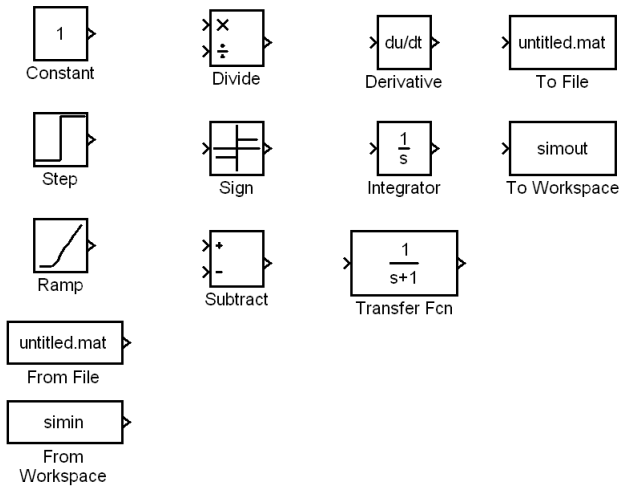

Figura 3.2: Outros blocos de uso recorrente

3.3 Virtual Reality Toolbox (VRT)

3.3.1 Introdução à Virtual Reality Toolbox (VRT)[1]

A VRT é um suplemento do software Matlab® que comporta funcionalidades ao nível da criação e simulação de ambientes em realidade virtual, fazendo uso de uma interligação entre os blocos desta *toolbox* e o Matlab® e/ou o Simulink® . Feita a interligação entre os modelos desenvolvidos em Simulink® (ou projectados com base no Matlab®), e os mundos virtuais criados recorrendo à VRT obtemos uma solução interactiva que produz resultados em tempo real e de forma gráfica.

[1] Nas versões mais actuais (a partir da versão R2008) a Virtual Reality Toolbox passou a designar-se por Simulink 3D Animation Toolbox

A criação de mundos virtuais é realizada através da linguagem de programação VRML (Virtual Reality Modeling Language), a qual vai ser discutida no ponto 3.3.3. O processo de criação de ambientes virtuais é facilitado através de um editor de VRML e de um visualizador de ficheiros de extensão *.wrl* apresentado no ponto 3.3.6. A ferramenta fornecida para edição de cenas tridimensionais denomina-se de V-Realm Builder 2.0 (ver descrição no ponto 3.3.5). Esta trata-se de uma interface gráfica especialmente orientada para edição e criação de cenas 3D, tornando todo este processo realizável em alto nível e com um elevado grau de abstração.

3.3.2 Linguagem de Programação VRML - Vista Geral

A linguagem de programação VRML (Virtual Reality Modeling Language) é uma tecnologia relativamente recente que remonta ao ano de 1995, quando inicialmente era chamada de *Virtual Reality Markup Language* (VRML1). A sua origem teve como principal alavanca a expansão da *World Wide Web* (www) e a necessidade de se criar uma norma única para ambientes 3D na *web*. Esta versão foi criada de forma a ser suportada por vários *browsers*, permitindo navegar num qualquer ambiente tridimensional imbutido numa página *web*.

No entanto, o VRML1 não possuia capacidades ao nível de interacção e animação dos objectos presentes no ambiente 3D. O standard VRML1 foi modificado, dando origem ao VRML 2.0. No ano de 1997, o VRML 2.0 foi adoptado como linguagem padrão para web 3D e passou a ser referenciado frequentemente como VRML97.

Os avanços da tecnologia quer ao nível de rapidez de processamento e comunicação quer ao nível de capacidade gráfica, conjuntamente com a flexibilidade da linguagem VRML, tornou esta ferramenta de desenho computarizado bastante popular no domínio dos videojogos e da arte digital.

Relativamente às características da linguagem de programação VRML, esta assemelha-se ao *HyperText Markup Language* (HTML), pois os mundos 3D são também definidos através de ficheiros de texto (ver código 3.1) e orientados ao *www*, mas apresentando um resultado gráfico final diferente, que pode ser observado na figura 3.3.

Algorithm 3.1 Exemplo de um ficheiro VRML num editor de texto

```
#VRML V2.0 utf8
WorldInfo {
        title "Bouncing Ball"}
Viewpoint {
        position 0 5 30
        description "Side View"}
DEF Floor Transform {
        geometry Box{
        size 6 0.2 6}}
DEF Ball Transform {
        translation 0 10 0
        children Shape {
        appearance Appearance{
        material Material{
        diffuseColor 1 0 0
}}
        geometry Sphere{
}}}
```

As diferenças mais relevantes são experienciáveis através de um *browser* com recurso a um *plug-in* que suporte VRML ou executando um VRML *Viewer*. Entre as capacidades desta ferramenta destacam-se a possibilidade de navegar tridimensionalmente por todo o mundo virtual , visualizar os objectos de pontos de vista pré-definidos ou "voar" em torno dos mesmos e entrar em superfícies/objectos que permitam esse tipo de movimentações [25]. Para além deste tipo de interacção, a versão VRML 2.0 permite incutir comportamentos dinâmicos aos objectos, dependendo estes de determina-

das acções executadas no mundo virtual, através da execução de *scripts* ou simplesmente despoletados pelo decorrer do tempo.

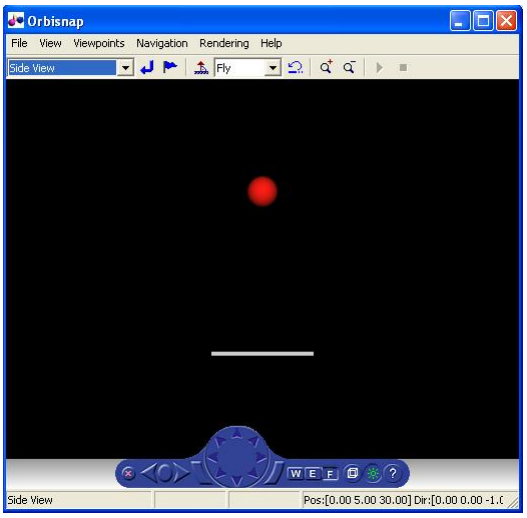

Figura 3.3: Resultado visual produzido pelo ficheiro VRML presente no algoritmo 3.1

Desde a criação do VRML 2.0 desenvolveram-se ferramentas que permitem a reprodução de ambientes virtuais facilmente. Entre estas ferramentas podem encontrar-se programas conversores de ficheiros noutros formatos gráficos para VRML, ferramentas de CAD (*Computer Assisted Design*) que permitem exportar os seus ficheiros para VRML 2.0 e ainda uma vasta gama de aplicações de edição e criação de ficheiros em VRML.

Como qualquer boa ferramenta de software, também esta tem vindo a ser melhorada devido aos progressos a nível de hardware, não deixando de parte a importância da compatibilidade com as versões anteriores. Deste modo, a sucessora da linguagem VRML 2.0 está já acessível, depois de o *Web 3D Consortium* ter lançado o novo *standard* denominado de X3D (eXtensible 3D).[20]

3.3.3 Características da VRML

O uso da linguagem VRML por parte da VRT do Matlab® pode trazer algumas complicações a nível de percepção dos ambientes 3D e da dinâmica presente. Este facto pode dever-se desconhecimento de algumas das características básicas da programação em VRML 2.0. Por este motivo, abordam-se nos parágrafos seguintes alguns dos conceitos implementados pela VRML que acabam por ser de extrema relevância quando tentamos integrar o Matlab® com estes mundos virtuais.

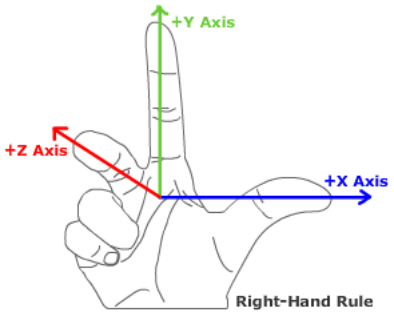

Figura 3.4: Sistemas de eixos VRML (retirado de[1])

Ao contrário do sistema de coordenadas usado por grande parte das aplicações integrantes do software Matlab®, a VRT, por consequência directa do uso da VRML, orienta-se por um sistema de

eixos cartesiano dado pela regra da mão direita (ver figura 3.4). Com este sistema de coordenadas os eixos x, y e z definem respectivamente o comprimento, a altura e a profundidade de qualquer objecto no ambiente virtual. Pela representação na figura 3.4 é visível que o sentido positivo da variação na posição é dado pela direcção para a qual apontam os dedos. Assim, a variação é positiva em x, y e z , respectivamente, para a direita, para cima e na direcção do utilizador.

Outra particularidade da representação em VRML prende-se com o facto de os ângulos de rotação serem igualmente realizados segundo a regra da mão direita. Na prática equivale a afirmar que o sentido positivo de rotação em cada um dos eixos coincide com o sentido anti-horário, pois fazendo apontar o dedo polegar no sentido positivo de cada um dos três eixos, o ângulo e sentido de curvatura é dado pelos restantes quatro dedos como mostra a figura 3.5.

Figura 3.5: Regra da mão direita para determinação do ângulo de rotação

Quando abrimos um ficheiro com extensão *.wrl* num editor de VRML ou através de um editor de texto, este apresenta uma estrutura hierárquica composta por vários nós. Estes nós são depois subdivididos, apresentando desse modo características específicas de cada um em várias camadas.

Uma dessas subdivisões é feita através dos nós tipo *child*, os quais representam um objecto ou uma forma que se encontra acoplada a outros objectos e formas de maior importância. A grande vantagem em definir um objecto através de um nó deste tipo é permitir que esta forma mantenha o seu posicionamento relativamente ao objecto-pai (*parent object*). Neste ponto há que ter em atenção alguns aspectos de posicionamento, pois para o objecto-filho (*child object*) a origem do sistema de coordenadas é ditado pelo objecto-pai e, assim sendo, a dinâmica associada ao objecto-pai afecta sempre o objecto-filho.

No que a medidas diz respeito, a VRT usa o sistema de medidas definido pela VRML, ou seja, todos os ângulos são medidos em radianos e todas as distâncias e dimensões de objectos são definidas em metros.

3.3.4 Ficheiros e Extensão .wrl

A extensão usada para ficheiros escritos em VRML é a extensão *.wrl*. Ficheiros deste tipo podem ser editados, tal como foi já mencionado anteriormente, através de um simples editor de texto (algoritmo 3.1), caso sejamos programadores experientes com este tipo de linguagem, ou com uma ferramenta de edição/criação de ficheiros *.wrl*, se pelo contrário somos iniciantes nas lides das criações 3D. A sintaxe de programação em *Virtual Reality Modeling Language,* apesar de ser relativamente simples, segue algumas regras de estrutura que, ao não serem respeitadas, resultam em ficheiros corrompidos e não funcionais. Por esse motivo, o uso de uma ferramenta de criação e edição de mundos virtuais em VRML 2.0 será o adequado para o primeiro contacto com este tipo de tecnologia, mesmo apesar de os melhores mundos virtuais ainda serem criados a partir dos editores de texto.

Qualquer ficheiro VRML, por mais simples que seja, deve sempre apresentar na sua primeira linha um cabeçalho de formato fixo. A responsabilidade deste cabeçalho é informar o browser ou o visualizador em utilização qual a interpretação a dar ao código que o sucede. Deste modo, a primeira linha de um ficheiro com extensão *.wrl* deve coincidir com:

```
#VRML V2.0 utf8
```

Estas directivas vão permitir saber à aplicação para visualização de mundos virtuais que está prestes a lidar com um ficheiro do tipo VRML ($\#VRML$). Seguidamente é passada a versão em que se encontra escrito o código, neste caso a versão 2 ($V2.0$), e finalmente qual a codificação de caracteres usada ($utf8$)[2]. Todas as linhas consequentes serão interpretadas como código VRML desde

[2] É um tipo de codificação que requer até 4 bytes, permitindo codificar caracteres diferentes dos existentes no código ASCII. Esta codificação faz parte da uniformização dos sistemas informáticos dado a sua capacidade de exprimir todos os caracteres da codificação Unicode e ser compatível com os códigos ASCII.

que não estejam precedidas pelo símbolo cardinal (#), pois este define quais as linhas de comentário à excepção da primeira linha de cada ficheiro.

Um ficheiro .wrl pode apresentar até 54 tipos diferentes de nós, estando estes distribuídos hierarquicamente numa estrutura em árvore como demonstra janela mais à esquerda da figura 3.6. Cada nó desta árvore representa uma característica, um objecto ou uma funcionalidade do ambiente virtual. Entre as funcionalidades disponibilizadas pelos 54 tipos de nós apresentam-se a seguir as mais importantes:

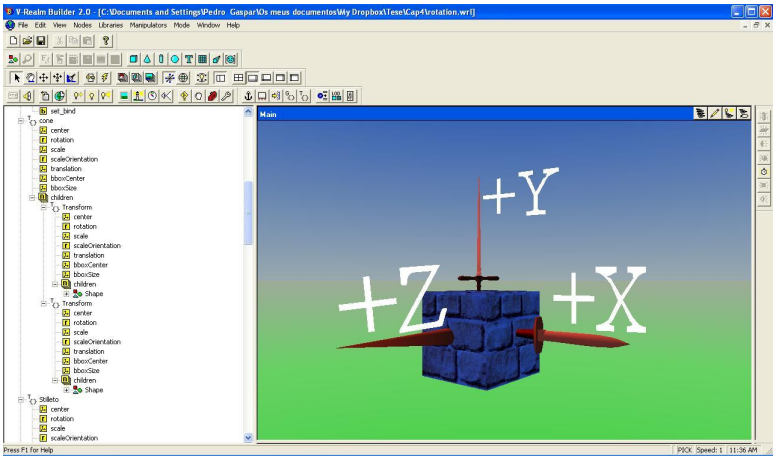

Figura 3.6: Estrutura em árvore, hierarquicamente dividida

Transform Node/Group Node

Permite definir propriedades como a posição, a orientação, a escala, a rotação, a translação e disponibiliza ainda um campo do tipo *children*, o qual pode ser utilizado para acoplar um ou mais objectos-filho a esta árvore.

Material Node

Define o tipo de material pretendido para constituir o nosso objecto.

Fog Node

Nó responsável por modificar as propriedades ópticas da cena virtual.

Directional/Point/Spot Light Node

Representa a luz no mundo virtual, dando a possibilidade de definir características como a cor, a intensidade, a direcção, a largura do feixe, a atenuação ou posição.

Texture Node

Caracteriza o tipo de textura que desejamos para o objecto.

ViewPoint Node

Pré-define uma posição ou ângulo de visão sobre o ambiente virtual.

Box/Sphere/Cone Node

Representa um volume com uma determinada forma no ambiente virtual.

Extrusion Node

Permite moldar um volume através do reposicionamento dos seus vértices originais ou pela criação de novos vértices.

Indexed Face Node

Ao contrário do *Extrusion Node*, este tipo de nó disponibiliza funções de modificação de superfícies planas utilizando o mesmo conceito referido, ou seja, ajustando a posição dos vértices que compõe a forma original.

Shape Node

É dentro deste ramo que podemos colocar um dos três últimos nós descritos.

Qualquer nó presente numa árvore VRML pode ser renomeado através da palavra chave *DEF*. Esta directiva torna-se importante na medida em que somente os nós marcados por ela tornam acessíveis os seus campos pelo próprio código VRML ou por qualquer aplicação externa que interaja com estes. Nos pontos 3.3.7 e 3.3.5 serão apresentados alguns casos de utilização não explícita desta palavra-chave e a sua importância para a interacção do VRML com os modelos a criados em Simulink®.

Chegados a este ponto, é já possível ter uma impressão de como tudo funciona na programação em VRML. No entanto, há ainda variadas características e funções extremamente úteis que devem ser conhecidas para se obterem bons resultados na criação de ambientes 3D. As funções às quais me refiro estão relacionadas com as propriedades de cada tipo de nó. Estas propriedades podem ser conhecidas através da leitura das especificações da linguagem VRML ou simplesmente visualizando e analisando os nós de ficheiros *.wrl* já existentes.

Cada tipo de nó tem os seus próprios campos, sendo estes responsáveis por descrever determinadas características. Analogamente à grande parte das linguagens de programação, também o VRML faz distinção entre tipo de variáveis, o que no seu caso específico equivale dizer que os campos de cada nó aceitam valores específicos. Estes valores podem ser inteiros, *floats*, caracteres, boleanos e *arrays* de cada um dos tipos referidos anteriormente. Na tabela 3.1 apresentam-se individualmente as possibilidades de valores que os vários campos podem tomar:

Para além dos diferentes tipos de dados que os campos de cada nó exprimem, a VRML 2.0 introduz igualmente o conceito de classes de dados. Existem no máximo quatro classes de dados por cada nó: *field*, *exposedField*, *eventIn* e *eventOut*. As diferentes classes permitem definir comportamentos para os nós, a forma como estes interagem com outros nós ou objectos externos e a maneira como estes são armazenados na memória dos computadores [20]. Na tabela 3.2 (apresentada mais adiante) são explicitadas as particularidades de cada uma destas classes de dados.

De modo a perceber-se melhor quais as possibilidades de uso das quatro classes de dados definidas pela norma VRML, listam-se em seguida com uma descrição mais completa:

eventIn

Alterar este tipo de nós implica ter um evento associado, pois este não é acessível a partir do próprio nó. Esta funcionalidade significa que um nó pode ter um evento *eventIn* associado sem este corresponder a um campo específico desse nó. Um exemplo comum, será o uso de um evento de entrada (*eventIn*) num dos sub-campos de um nó do tipo *Group* ou *Transform*. Assim, quando este evento é recebido de um objecto externo, os nós passados são adicionados à lista desse nó ou sub-nó do tipo *Group/Transform*.

eventOut

Os eventos do tipo *eventOut*, associados a um determinado campo de um nó, são enviados para nós do tipo *eventIn* sempre que existe modificação no seu valor.

field

Os campos são privados dos nós que os detêm e podem tomar os valores mencionados anteriormente se estes receberem um *eventIn* correspondente. De outra forma, não é possível alterar o valor desta classe de dados.

exposedField

É a classe de dados que mais funcionalidades apresenta ao nível do VRML. O facto de funcionar com os eventos do tipo *eventIn* e *eventOut* permite definir como estes se vão comportar quando funcionam cooperativamente. Para todas as classes *exposedField*, a ocorrência de um evento provoca a mudança do valor do campo, consequentemente, uma alteração na aparência da cena virtual e com isso a geração de um *eventOut* com o novo valor do campo. Isto autoriza a encadeação de eventos através de vários nós [20].

Tipo de dados VRML	Descrição
SFBool	Valor boleano: "True" ou "False"
SFFloat	Valor de 32 bits em vírgula flutuante (floating point)
SFInt32	Um valor inteiro com sinal em 32 bits SFInt32_value = floor(double_value)
SFTime	Define um valor de tempo relativo ou absoluto
SFVec2f	Vector de 2 valores em vírgula flutuante usados normalmente para definir coordenadas 2-D (definição de texturas)
SFVec3f	Vector de 3 valores em vírgula flutuante usados frequentemente para representar coordenadas 3-D
SFColor	Vector tridimensional em vírgula flutuante usados para especificar a codificação de cores RGB
SFRotation	Vector de 4 valores em vírgula flutuante usados para definir as coordenadas de rotação (x,y,z) de um eixo e o ângulo de rotação sobre esse eixo
SFImage	Array bidimensional representado por uma sequência números em vírgula flutuante
SFString	String codificada em UTF-8, compatível com o código ASCII Permite o uso de caracteres Unicode
SFNode	Recipiente para um nó VRML
MFFloat	Vector de valores SFFloat
MFInt32	Vector de valores SFInt32
MFVec2f	Vector de valores SFVec2f
MFVec3f	Vector de valores SFVec3f
MFColor	Vector de valores SFColor
MFRotation	Vector de valores SFRotation
MFString	Vector de valores SFString
MFNode	Vector de valores SFNode

Tabela 3.1: Tipos de dados VRML (adaptado de [20])

Classes de Dados VRML	Descrição
eventIn	Um evento que pode ser recebido pelo nó
eventOut	Um evento que pode ser enviado pelo nó
field	Membro privado de um nó, com informação relativa ao nó
exposedField	Campo público de um nó, contendo dados do nó

Tabela 3.2: Classes de dados em VRML (adaptado de [20])

3.3.5 V-Realm Builder 2.0 - Editor de VRML

O editor de ficheiros VRML disponibilizado pela VRT do Matlab® é uma ferramenta desenvolvida pela Humusoft® e construída sobre uma rede virtual de ficheiros. Um dos objectivos deste poderoso pacote de construção de cenas 3D é disponibilizar funções de modelação que possam servir programadores experientes mas também novos utilizadores desta tecnologia. No entanto, o cerne desta

ferramenta prende-se com a realização de ambientes virtuais portáveis e facilmente carregados e visualizados num browser com acesso à internet. Por este motivo, a aplicação V-Realm Builder 2.0 gera ficheiros VRML altamente compactados e com código extremamente eficiente, recorrendo a várias técnicas recursivas de programação e ao uso de polígonos primitivos para a construcção de objectos mais complexos.

O V-Realm Builder 2.0 apresenta uma interface gráfica de utilizador simples e intuitiva como é visível na figura 3.7. A interface contém três áreas de acção, designadas por área A, B e C. A área representada por A disponibiliza todas as funções possíveis de usar no mundo virtual, permitindo adicionar objectos, escolher o modo de visualização e inserir algumas propriedades de nível óptico ou cromático; na área B pode observar-se a árvore com diferentes nós que é construída acrescentando objectos e características ao ambiente 3D. Todas as acções realizadas nas áreas A e B são reflectidas visualmente na área C.

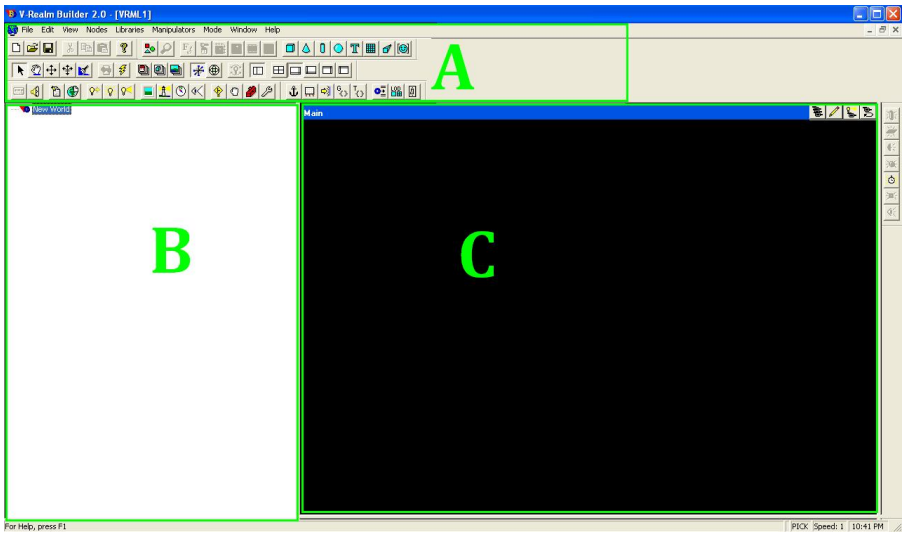

Figura 3.7: Interface de utilizador da aplicação V-Realm Builder 2.0

A área representada por A na figura anterior pode ser decomposta em pequenas barras de ferramentas. Estas estão assinaladas na figura 3.8.

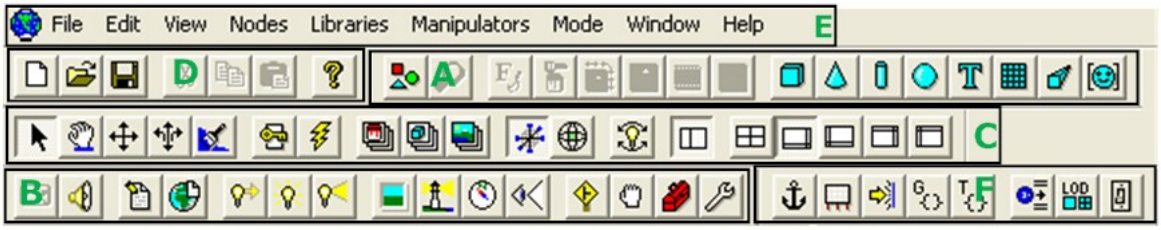

Figura 3.8: Barras de ferramentas do editor V-Realm Builder

A barra de ferramentas assinalada por A constitui um conjunto de formas que podemos adicionar ao nosso ambiente virtual.

Através deste bloco de funções, discriminado na ilustração 3.9 (ver página seguinte), podemos facilmente adicionar uma forma cúbica (botão nº 9), um cone (10), um cilindro (11), uma esfera (12) ou simplesmente texto (13). Além dos sólidos geométricos, podem ser inseridas figuras geométricas simples (1), escolher um tipo de letra (3), importar um determinado tipo de material (4) ou importar texturas: predefinidas (5), de imagens (6), de um filme (7) ou de um pixel (8).

Os três blocos mais à direita da figura 3.9 permitem inserir uma grelha de elevação (14), uma extrusão (15) ou um nó do tipo indexed face set (16). A grelha de elevação (*elevation grid*), possibilita a inserção de uma área rectangular na qual podemos criar relevos. Este tipo de forma é ideal, por

exemplo, para a criação de montanhas, vales, lombas ou descontinuidades no solo. O botão 15 adiciona ao mundo virtual formas cuja secção transversal é fixa, permitindo obter formas extremamente complexas. Finalmente, o botão 16 acrescenta um nó do tipo *indexed face set*. Esta funcionalidade é uma escolha recorrente quando se pretendem criar objectos constituídos por malhas.

Figura 3.9: Barra de ferramentas do editor: formas geométricas

O conjunto de blocos B, na figura 3.8, representa um conjunto de funcionalidades de nível óptico, informativo, recursivo e sonoro, passíveis de acrescentar à cena 3D que pretendemos implementar. Como é visível, os botões encontram-se devidamente agrupados para facilitar a percepção das funções que estes desempenham. O grupo mais à esquerda, botões 1 e 2, permite inserir no ambiente virtual , respectivamente, uma textura de vídeo e/ou um som. Apresentam-se depois os botões 3 e 4 que, quando seleccionados, adicionam à estrutura em árvore um *script* ou informação sobre o ambiente 3D. Para se inserirem focos luminosos existem os botões 5, 6 e 7, inserindo respectivamente: luz direccional, fonte de luz estática e ainda uma luz que radia omni-direccionalmente. Outras propriedades ópticas são conseguidas com as funções numeradas de 8 a 11 da figura 3.10. O botão 8 adiciona um fundo pré-definido ao ambiente virtual. O botão 9 serve para inserir nevoeiro na cena, permitindo modificar características como o alcance de visibilidade, a cor e o tipo de nevoeiro (linear ou exponencial). A função 10 apresenta uma funcionalidade pouco vulgar, prendendo-se o seu objectivo com o modo de navegação que é feito no mundo virtual. Esta navegação está limitada aos tipos definidos pelo VRML: *fly, examine, walk e none*. Clicando no botão 11 é permitido editar e/ou criar pontos de vista diferentes para o ambiente virtual projectado.

Aos quatro botões que se situam mais à direita da imagem 3.10 estão atribuídas funcionalidades relacionadas com recursividade, ou seja, são estes nós que permitem utilizar outros nós já criados na estrutura do ambiente virtual, evitando assim copiar extensivamente todos os campos e seus conteúdos. Este tipo de nós são os responsáveis por gerar ficheiros VRML tão compactos e portáteis, e o seu uso implica apenas que se aplique a directiva DEF (ou se defina um nome) no nó que se pretende reutilizar. Dentro destas funções enquadram-se respectivamente, os botões Route (12), Use (13), PROTO (14) e PROTO Instance (15).

Figura 3.10: Barra de ferramentas de botões comuns

Na figura 3.11 está apresentada a área C, marcada anteriormente na ilustração 3.8 . Aqui podemos encontrar opções de interacção com a cena 3D (botões 1, 2, 3, 4 e 5), modos de teste (7) ou de animação (6) e ainda bibliotecas de materiais (8), texturas (10) e alguns objectos (9). O primeiro grupo de opções dá ao utilizador a possibilidade de, respectivamente, seleccionar objectos (1) e movê-los dentro do ambiente virtual, rodar todo o mundo virtual e seus objectos segundo todas as direcções (2), mover-se horizontalmente e verticalmente na cena com total liberdade (3), percorrer a cena em profundidade e horizontalmente mantendo fixa a coordenada y e com total flexibilidade para rodar e recuar (4) e finalmente circular no ambiente no modo de cor (5).

Com o segundo conjunto de opções é-nos dada a possibilidade de animar os objectos inseridos no ambiente virtual e entrar no modo de teste para a cena em execução. A primeira opção, na base da interface, edita os movimentos dos objectos *frame* por *frame*, tirando partido da base de tempo que é apresentada assim que activamos este modo. Na segunda opção entramos num modo de teste que permite a execução e visualização das animações criadas no editor e em tempo real.

Relativamente às bibliotecas já referidas, o editor V-Realm Builder traz consigo um conjunto de ficheiros que revela muita utilidade quando pretendemos tornar os nossos mundos virtuais mais realistas. Essas características são acessíveis a partir do grupo de três botões numerados de 8 a 10 na figura 3.11. O primeiro (8) destes botões apresenta uma panóplia de escolhas em termos de conjunto cor-material. O botão seguinte (9) permite seleccionar vários objectos de diferentes categorias como edifícios, paisagens, armas, veículos ou até mesmo orgãos do corpo humano. Por fim, o botão (10), podemos escolher diferentes texturas para os objectos em cena.

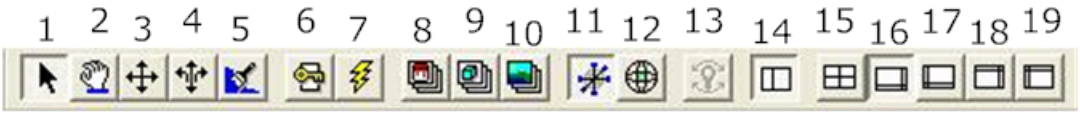

Figura 3.11: Barra de ferramentas de modos de navegação, visualização e modificação de objectos

Ainda relativamente à barra de ferramentas na figura 3.11 , são apresentadas algumas opções relacionadas com a forma como o utilizador do editor V-Realm Builder pretende visualizar a cena 3D que está a criar (funções numeradas de 11 a 19). Os botões *toggle Universal Manipulator* (11) e *Centerball Manipulator* (12) deste conjunto de botões permitem manipular os objectos seleccionados na cena aquando da sua activação. O *Universal Manipulator* funciona como expansor do objecto seleccionado em qualquer dos 3 eixos. Por outro lado, o *Centerball Manipulator* posiciona uma esfera em torno do objecto correntemente seleccionado dando a possibilidade de rodar o objecto.

Seguidamente, apresenta-se a função (13), cujo objectivo é apresentar ao utilizador onde estão posicionadas as luzes inseridas anteriormente na cena, autorizando-o a reposicioná-las e a modificar a direcção de incidência.

Mais à direita na barra de ferramentas acima (figura 3.11), encontram-se seis botões que seleccionam o modo de visualização da janela da interface gráfica do editor. O (14) activa e desactiva a visualização simultânea da árvore de nós que vai sendo construída e da cena 3D que vai sendo montada. Os restantes cinco ajustam a janela destinada à observação do ambiente virtual de acordo com vários pontos de vista. Assim, da função 15 à 19, podemos observar a cena de cinco maneiras distintas: ver simultaneamente a vista principal, de topo, da direita e de frente; ver apenas a vista principal,observar a cena de cima, ver no plano XOY ou apenas a vista frontal e rectaguarda.

Na área F apresentada na imagem 3.8 estão englobados alguns dos nós mais importantes ao nível deste editor de ficheiros VRML. O V-Realm Builder apresenta nesta barra de ferramentas (ver figura 3.12) um grupo de funções que possibilita ao utilizador a adição de nós especializados para a introdução de objectos geométricos e nós úteis para o carregamento de objectos de um outro local ou um outro mundo. Para além destas particularidades, são estes poderosos nós a base da construção de ambientes virtuais interactivos. Assim, da esquerda para a direita é possível inserir na estrutura do ambiente virtual nós do tipo: *anchor* (1), *billboard* (2), *collision* (3), *group* (4), *transform* (5), *inline* (6), LOD (*Level of Detail*) (7) e *switch* (8).

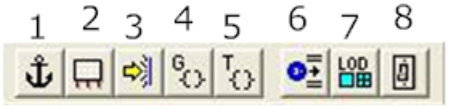

Figura 3.12: Barra de ferramentas de grupos de nós

De modo a permitir a alteração do estado dinâmico dos objectos no ambiente virtual por acção do utilizador ou simplesmente pelo decorrer do tempo, o V-Realm Builder disponibiliza um conjunto de botões que simulam a implementação se sensores. Estas funcionalidades estão acessíveis pela barra de ferramentas na figura 3.13. Os sensores fornecidos pelo editor de VRML são: sensor cilíndrico (1), plano (2), de proximidade (3), esférico (4), de tempo (5), de toque (6) e de visibilidade (7).

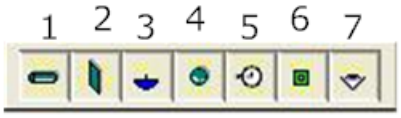

Figura 3.13: Barra de ferramentas de sensores

Na janela principal de visualização do mundo virtual é igualmente fornecida uma barra de ferramentas. As funcionalidades que esta disponibiliza relacionam-se com a navegação do utilizador na cena 3D. Como é apresentado na figura 3.14, este utensílio permite navegar para diferentes pontos de vista (1); escolher o modo de renderização do ambiente virtual (2); activar/desactivar a luz frontal à cena (3) ou voltar à posição predefinida (4).

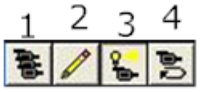

Figura 3.14: Barra de ferramentas da janela principal de visualização

Com o menu de opções na figura 3.15 facilmente criamos um novo ficheiro VRML (1), abrimos um já existente (2) ou gravamos as alterações feitas (3). A acrescentar a este menu *standard* de qualquer interface gráfico de utilizador, juntam-se ainda os botões de *cut* (4), *copy* (5) e *paste* (6) que se revelam extremamente úteis para copiar nós de um ficheiro para outro ou para simplesmente reposicioná-lo na árvore do ficheiro VRML. A finalizar o segmento de funções na figura abaixo (ver figura 3.15), observa-se um botão (7) cujo símbolo é ?, o qual fornece um guia ao utilizador com as informações mais importantes sobre como usar esta ferramenta.

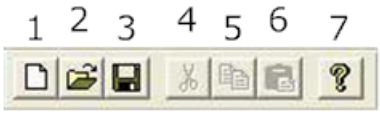

Figura 3.15: Standard menu

Finalizando a descrição do editor de ficheiros VRML, V-Realm Builder 2.0, uma nota para a possibilidade de realizar todas as acções mencionadas anteriormente a partir dos menus *pop-up* na parte superior da aplicação (ver área E na figura 3.8) como exemplifica a figura 3.16.

Figura 3.16: Menu pop-up de contexto

3.3.6 Visualizador Orbisnap e Plug-in VRML

Existem vários aplicativos de software para a visualização de mundos virtuais no formato VRML. Neste aspecto o Matlab® aconselha dois produtos tendo em conta a finalidade dos ambientes virtuais que vão sendo criados. A ferramenta Orbisnap, incluída no pacote de instalação da VRT, é uma das aplicações aconselhadas e possibilita a visualização e navegação em ficheiros de realidade virtual.

Apesar de ser distribuida conjuntamente com o software da MathWorks®, este VRML Viewer consegue ter um funcionamento independente do Simulink® e do Matlab®. Assim, podemos usar

o Orbisnap para visualizar um ambiente virtual previamente animado ou se preferirmos, usá-lo para visualização de um ficheiro *.wrl* que se encontra controlado por um modelo em Simulink®. No primeiro caso, não necessitamos de acesso ao software Matlab® e o Orbisnap funciona como uma aplicação autofuncional (*stand-alone application*). Já no segundo caso, a animação do ambiente 3D é gerada por um modelo matemático desenvolvido no Simulink®, sendo por isso necessária a execução simultânea das duas ferramentas de software.

Uma das características mais notáveis deste visualizador de ficheiros VRML é o facto de ser uma ferramenta gratuita, ou seja, não é necessária qualquer licença para a executar. O seu uso implica apenas o seu download a partir da página `www.orbisnap.com/download.html` e a execução do processo de instalação.

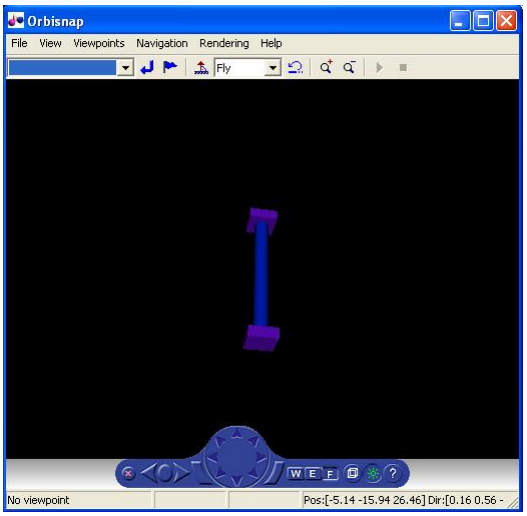

Figura 3.17: Orbisnap: VRML Viewer

Depois de instalada, a aplicação Orbisnap apresenta uma interface gráfica de utilizador com o aspecto apresentado na figura 3.17.

A interface gráfica de utilizador disponibilizada pode caracterizar-se por quatro áreas distintas. Na parte superior é apresentada uma barra de menus com as seguintes opções:

File

- *Open* - Apresenta um browser com o qual podemos localizar os ficheiros .wrl que pretendemos visualizar.
- *Connect to server* - Possibilita a conexão com a Virtual Reality Toolbox Server (VRTS) através do endereço IP da máquina que está a executar o VRTS e da porta em que esta a escutar.
- *Reload* - Recarrega o ambiente virtual actual. Caso tenham sido criados novos pontos de vista (*Viewpoints*) e não tenham sido gravados, estes perder-se-ão.
- *Save as* - Permite guardar o ambiente 3D.
- *Close* - Termina o visualizador Orbisnap.

View

- *Toolbar* - Activa/Desactiva a visualização do painel de controlo na parte inferior da janela.
- *Status Bar* - Activa/Desactiva a barra de estado na base da janela do visualizador. Nesta barra é disponibilizada informação sobre a posição actual, tempo de simulação, modo de navegação, a posição da camera e a direcção.
- *Navigation Zones* - Activa/Desactiva as zonas de navegação no mundo virtual.

- *Navigation Panel* - Controla a visualização e a activação do painel de navegação.
- *Zoom In/Out* - Amplia ou reduz a visualização do ambiente virtual.
- *Normal (100%)* - Regressa ao ponto de vista inicial.
- *Fullscreen Mode* - Apresenta o mundo virtual em ecrã completo.

Viewpoints

Faz a gestão dos pontos de vistas disponíveis, permitindo ainda adicionar novos viewpoints e remover outros.

Navigation

Permite escolher o método e a velocidade de navegação, voltar ao ponto de navegação inicial ou desfazer um movimento.

Rendering

Gere a renderização das cenas: define quais as características (como por exemplo a textura ou luminusidade) dos objectos que são processadas pelo visualizador.

Help

Disponibiliza uma janela para localização de ajuda para o Orbisnap.

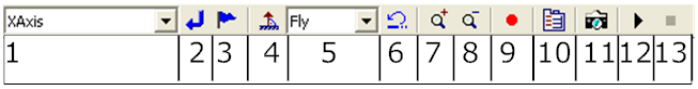

Figura 3.18: Barra de ferramentas do Orbisnap

Abaixo da barra de menu encontra-se a barra de ferramentas da aplicação. Tendo em conta a numeração dos botões na figura 3.18, esta barra oferece como funcionalidades: escolher o ponto de vista a partir da lista (1), retroceder para o ponto de vista anterior (2), criar um ponto de vista (3), endireitar a visualização (4), escolher um modo de navegação a partir da lista disponibilizada no botão 5 (*walk, fly e examine*), desfazer uma movimentação (6), ampliar (7) ou afastarmo-nos (8) da cena principal, gravar a animação do ambiente 3D em formato de vídeo (9), modificar os parâmetros do bloco "VR Sink" através da função 10, capturar um *frame* (11) do mundo virtual em execução e, finalmente, começar (12) ou parar (13) a simulação do ambiente virtual.

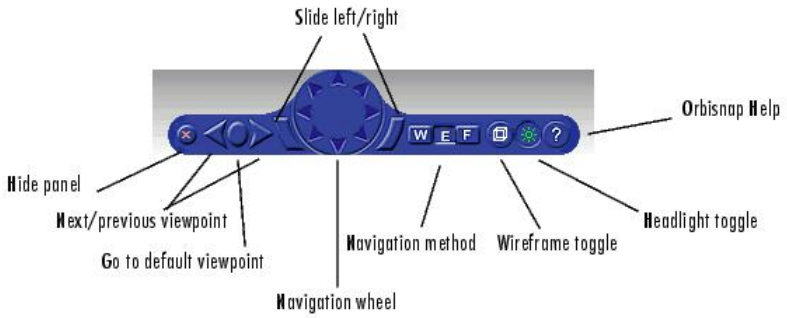

Figura 3.19: Painel de controlo da ferramenta Orbisnap (retirado de [20])

A área de maior proporção permite a visualização do mundo virtual que carregámos para o VRML Viewer. É nesta área que nos podemos movimentar utilizando o rato. De modo a obter total liberdade

de movimento, o Orbisnap disponibiliza ainda um painel de controlo na parte inferior da sua janela. Com este painel podemos realizar todos os tipos de movimentações na cena, ou seja, colocarmo-nos ou rodar sobre os três eixos, colocarmo-nos num determinado local no ambiente 3D ou mudar de ponto de vista. Na figura 3.19, está representado esse painel de controlo com as suas funcionalidades:

Este painel acaba por ser uma mais valia em toda a interface pois possibilita a realização das funções mais importantes, e já descritas, a partir dele.

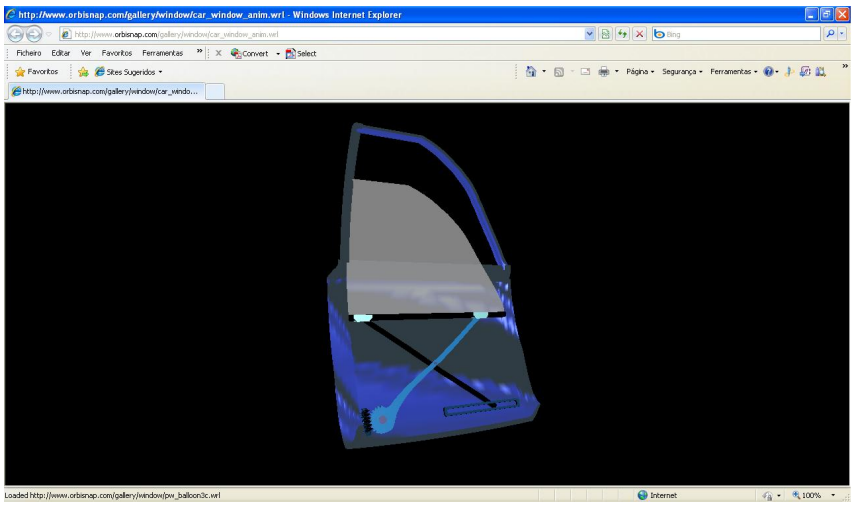

Figura 3.20: Visualização de um ficheiro VRML num *browser*: blaxxun Contact *plug-in*

Como havia mencionado, o Matlab® fornece duas opções na escolha do visualizador de mundos virtuais no formato *.wrl*. A segunda opção é a instalação de um *plug-in* VRML num *web browser*. Neste caso específico, o *plug-in* disponibilizado durante a instalação da VRT denomina-se por blaxxun Contact e está apenas orientado para os *browsers* Netscape® ou Microsoft® Internet Explorer e plataformas Windows. Dada a versão da VRT utilizada nesta tese, foi usada a versão 4.4 do *plug-in* da blaxxun Contact, obtendo-se desta forma a possibilidade de visualizar cenas virtuais em linguagem VRML no browser pré-definido (ver figura 3.20).

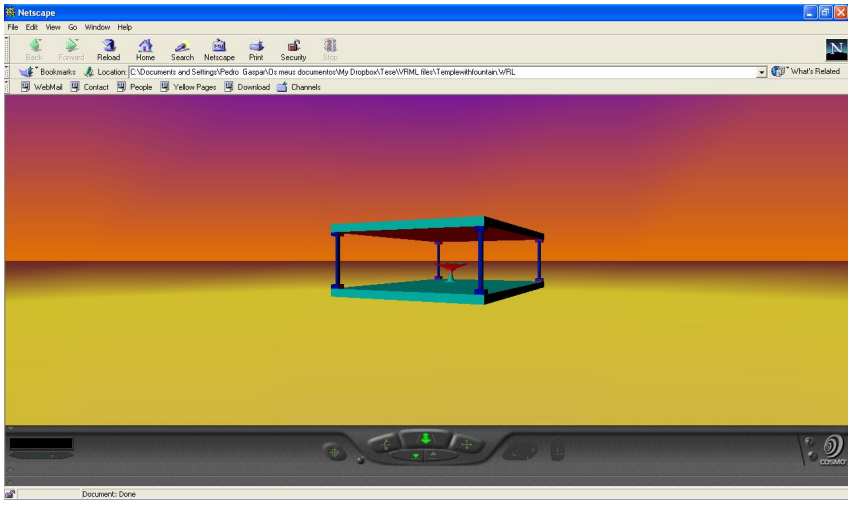

Figura 3.21: Visualização de um ficheiro *.wrl* num *browser*: Cosmo Player *plug-in*

Na figura 3.20 é visível a menor flexibilidade que existe quando vemos um ambiente 3D VRML num *web browser*, tirando partido do *plug-in* instalado para esse efeito. De facto, com o *plug-in* "aconselhado" na instalação de VRT perde-se alguma interactividade com os mundos virtuais e a navegação na cena fica limitada. Por esse motivo, testou-se a instalação de um outro *plug-in* indicado

para sistemas operativos Linux/UNIX, no browser Netscape® a correr no sistema operativo Windows XP. O resultado demonstra-se figura 3.21.

No caso apresentado pela figura 3.21 é conclusivo a melhor capacidade de movimento disponibilizada, pois com o plug-in Cosmo Player obtém-se um painel de controlo na base da janela que permite navegar segundo as opções já referidas para o visualizador Orbisnap.

3.3.7 Virtual Reality Toolbox - Blocos Funcionais

A caixa de ferramentas de realidade virtual fornecida pelo Matlab® permite animar mundos VRML a partir de modelos Simulink® ou através de *scripts* Matlab® que interagem com os ambientes virtuais e com os seus nós. Esta capacidade torna os mundos virtuais em verdadeiros ambientes de simulação e teste de sistemas de controlo, pois todo o comportamento dinâmico dos objectos é controlado de forma matemática e precisa, pelos dados e sinais gerados quer pelo Simulink® quer pelos comandos dos *scripts* criados.

Como foi referido no ponto 3.3.3 (na página 25), a VRML e o Matlab® apresentam algumas incompatibilidades em termos de especificações de funcionamento. Este facto implica que haja blocos de tratamento dos dados gerados pelo Simulink® e pelo Matlab®, de forma a obter-se informação utilizável pela VRT e consequentemente pelo ambiente descrito em VRML. Esses blocos são apresentados nos próximos parágrafos, seguidos de uma descrição funcional.

VR Sink

O bloco "VR Sink" apresenta-se como o principal bloco da consola de realidade virtual da aplicação Matlab®. Através deste bloco (ver figura 3.22) é possível escrever dados gerados por modelos de sistemas em Simulink® nos campos dos nós de um determinado ficheiro VRML. Na figura abaixo é apresentado um bloco "VR Sink" geral, ou seja, não tem ainda qualquer ambiente virtual acoplado e, por esse motivo, não apresenta qualquer porta de entrada.

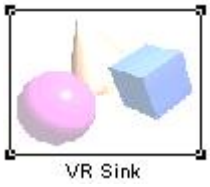

Figura 3.22: Bloco VR Sink

Para que seja possível a interacção de um modelo em Simulink® com o bloco acima é necessário fornecer alguns dados à caixa de diálogo visível na figura 3.23. Esta é-nos apresentada depois de um clique duplo sobre o bloco. Depois de executado este passo, a caixa de diálogo de parâmetros requer que sejam preenchidos ou seleccionados alguns campos de modo a produzir um resultado funcional. Existem quatro áreas de interacção na janela de diálogo do "VR Sink": a área *World Properties*, *Block Properties*, *VRML Tree* e a área informativa que apresenta o nome do bloco e uma breve descrição do seu modo de funcionamento.

A área à qual devemos dar inicialmente atenção é a *World Properties*, pois esta permite escolher, editar ou visualizar o ficheiro VRML que pretendemos acoplar ao nosso modelo Simulink®, definir algumas características relativas à forma como é acedido e acrescentar uma descrição. Relativamente às opções apresentadas por esta secção da janela de diálogo convém referir algumas particularidades. Na secção de escolha do *Source File*, é possível fornecer directamente o nome do ficheiro *.wrl* que pretendemos usar, mas com o cuidado de fornecer todo o caminho ou garantir que este se encontra no directório corrente de ambiente de trabalho. A secção *Output* permite assinalar se desejamos que o visualizador de ficheiros VRML seja aberto automaticamente e se pretendemos que o nosso ambiente virtual seja acessível a partir da internet.

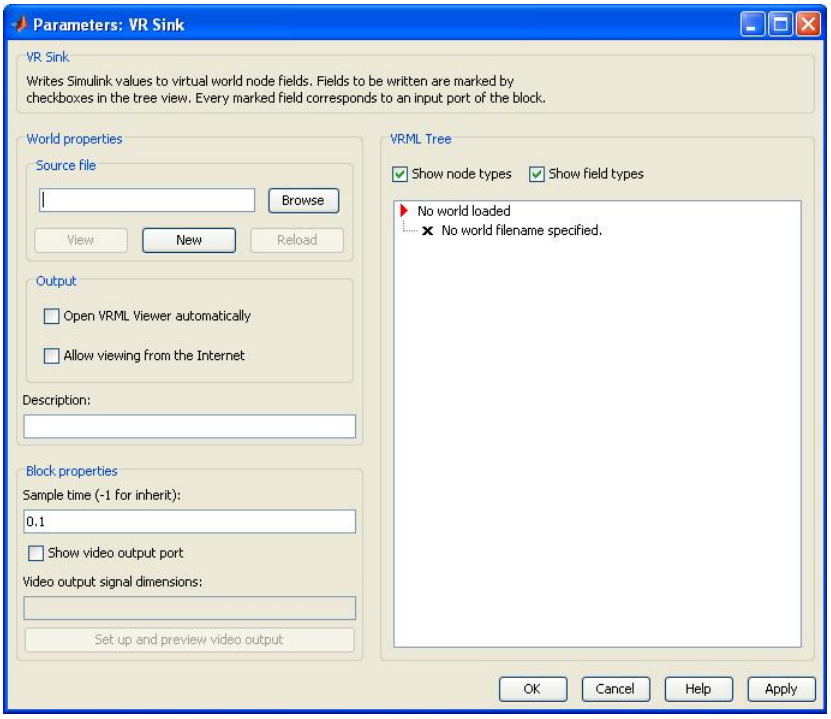

Figura 3.23: Block parameters dialog box

Abaixo da área *World Properties*, encontra-se uma zona destinada à especificação das características do bloco "VR Sink". É neste local da janela de diálogo que podemos definir o tempo de amostragem (*sample time*) e activar a porta de saída para vídeo com a resolução que se desejar.

À direita da caixa de diálogo da figura 3.23, temos acesso aos nós e aos respectivos campos do ficheiro VRML carregado, sendo ainda facultada informação sobre quais os tipos de nós e de campos que o compõem. A área *VRML Tree* é uma facilidade de acesso aos objectos (nós) do mundo virtual que pretendemos controlar a partir do modelo Simulink®. Assim, para cada objecto podemos controlar vários campos do nó que o representa. Apesar desta flexibilidade de acesso à estrutura dos ficheiros VRML, nem todos os nós têm permissões de escrita. Estas permissões são obtidas pelo próprio bloco "VR Sink" durante o carregamento do ficheiro *.wrl* e são assinaladas segundo um esquema de marcação. O esquema dita que os nós que possuem nomes, ou seja, antecedidos da directiva DEF (quando visualizados num editor de texto), são marcados por uma seta vermelha tornando-os acessíveis a partir do Matlab®. No caso de um nó não estar nomeado, mas tiver nomeado um seu nó tipo *child*, o esquema de marcação assinala também o nó tipo *parent* com uma seta vermelha. O acesso a cada campo através do Matlab® é realizado de forma independente, pois para cada campo de valor modificável existe uma *check-box*. Ao serem activadas, cada *check-box* origina uma porta de entrada de dados que deverão ser provenientes de um modelo realizado em Simulink®. Tipos de nós cujas propriedades e campos não permitem a modificação ou escrita pelo Matlab® são precedidos de um ponto azul. Ainda relativamente ao esquema de marcação usado na área *VRML Tree* da caixa de diálogo, podem existir nós não nomeados ou cujo tipo é diferente das classes de dados *eventIn* ou *exposedField*. Neste último caso particular é usado um ícone em forma de "x" para marcar a inacessibilidade desses nós e campos.

De modo a tornar este processo mais claro, ilustra-se na figura 3.24 uma caixa de diálogo devidamente preenchida e o respectivo resultado (ver figura 3.25) no bloco "VR Sink". Neste exemplo pretende-se controlar o movimento de translação de uma esfera e tornar acessível a exportação da dinâmica envolvida para um ficheiro de vídeo. De acordo com referido, carregou-se o modelo virtual de uma esfera entre dois planos, seleccionou-se o nó representativo da esfera na área *VRML Tree* da caixa de diálogo, expandiu-se esse nó e assinalou-se o campo "translation". Como era necessário fornecer acesso ao sinal de vídeo gerado pelo bloco "VR Sink", assinalou-se a opção "*Show video output*

port" na área *Block Properties*.

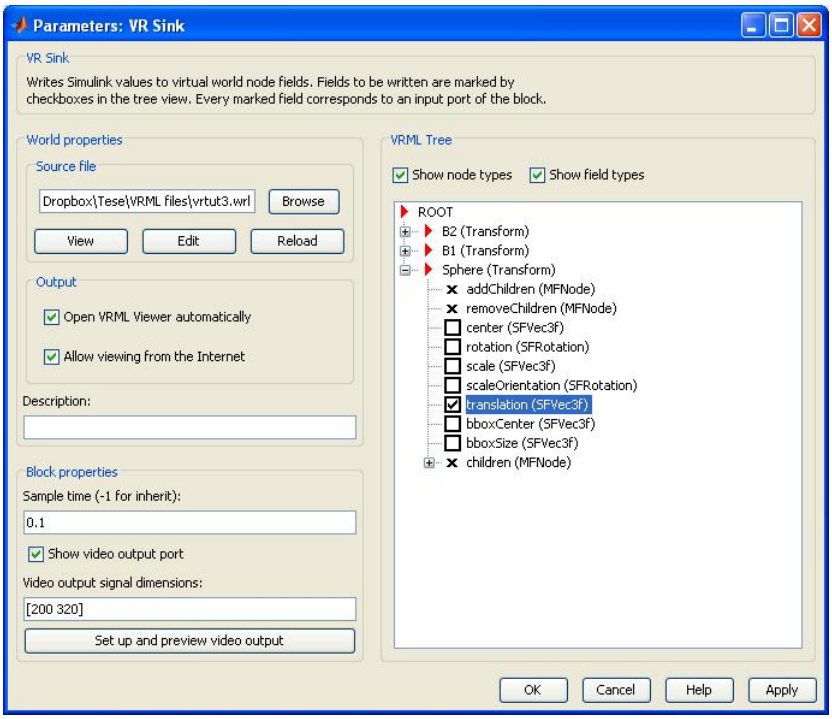

Figura 3.24: Preenchimento da dialog box do bloco VR Sink

Depois de aplicadas as definições através dos botões "OK" ou "*Apply*", a configuração final do bloco "VR Sink" (ver figura 3.25) apresentará duas portas: uma para entrada de dados responsáveis pelo controlo do movimento de translação da esfera e outra de saída do sinal de vídeo com a resolução definida anteriormente.

Figura 3.25: Bloco VR Sink: configuração final

VR Signal Expander

Outro bloco funcional da VRT é o "VR Signal Expander". O principal objectivo deste bloco é expandir vectores de dados no seu formato original para um formato compatível com os vectores dos campos VRML. Na figura 3.26 está presente o bloco já referido.

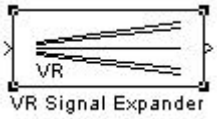

Figura 3.26: Bloco VR Signal Expander

O "VR Signal Expander" tem como entrada um vector de dados proveniente de um modelo de um sistema em Simulink® ou de uma matriz de dados carregada em *workspace*. A saída, por sua vez, é um vector de comprimento predefinido que pode utilizar todos ou apenas alguns dos valores do sinal de entrada. Neste último caso, os índices do vector de saída que não utilizam valores do sinal de entrada, são preenchidos através de um outro bloco denominado de "VR Placeholder", cujo funcionamento será adiante abordado. Este preenchimento é realizado pela interpretação dos parâmetros fornecidos na caixa de diálogo do bloco "VR Signal Expander" e segundo a sua máscara (ver figura 3.27).

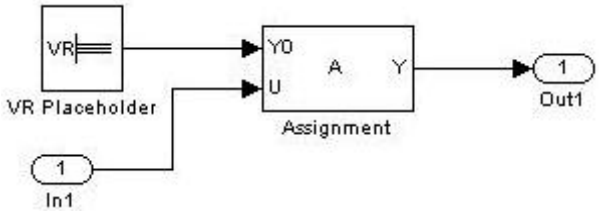

Figura 3.27: Máscara do bloco VR Signal Expander

O uso correcto deste bloco da VRT implica, tal como acontece com o "VR Sink", a definição de parâmetros do bloco de acordo com o sistema que com este interage. Esses valores são passados ao bloco através de uma caixa de diálogo (figura 3.28) aberta mediante um duplo clique.

Figura 3.28: Caixa de diálogo do VR Signal Expander

De acordo com o que foi visualizado no caso do bloco "VR Sink", também a caixa de diálogo do "VR Signal Expander" apresenta duas áreas distintas. Na parte superior da janela encontra-se o nome do bloco assim como uma a descrição do seu funcionamento e do que representa cada parâmetro a definir no bloco. No que a parâmetros concerne, a caixa de diálogo permite definir: primeiro, o comprimento do vector de saída (*Output width*), que deve ser definido de acordo com o campo do objecto que pretendemos controlar; depois devem ser definidos os índices do sinal de saída (*Output signal indices*), que indicam em que posição do vector de saída aparecem os sinais de entrada. Frequentemente pretendem-se controlar movimentos de rotação e/ou de translação o que significa que poderemos precisar, respectivamente, de um vector de saída do tipo $[x\, y\, z\, ¢]$ (*Output widht* $= 4$)

ou apenas $[x\,y\,z]$ (*Output width* = 3). Relativamente ao segundo parâmetro, a configuração poderá parecer um passo complexo, mas este passo prende-se com a incompatibilidade do sistema de eixos do Simulink® e da linguagem VRML (ponto 3.3.3 na página 25). No fundo, o que usualmente acontece é a troca da coordenada "y" no sistema de coordenadas adoptado pela VRML pela coordenada "z" do sistema de eixos imposto pelo Matlab®. Por exemplo, no caso de possuirmos em workspace um sinal de entrada ($v = [1\ 2\ 3; 1\ 3\ 3; 1\ 4\ 3]$) normalizado segundo o sistema de eixos do Matlab®, cujo conteúdo diz respeito às coordenadas por onde passa determinado objecto. Se pretendermos que o objecto virtual percorra as mesmas coordenadas, isto é, se movimente linearmente em profundidade e mantendo constantes as restantes coordenadas, então devemos preencher a caixa de diálogo do bloco "VR Signal Expander" como exemplifica a figura 3.29:

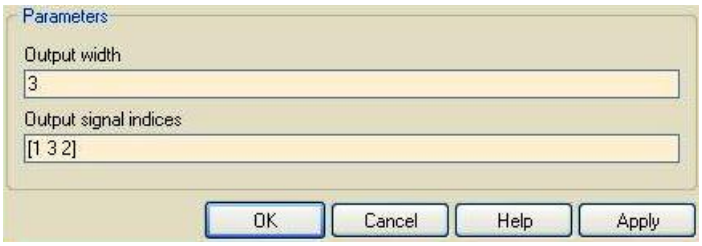

Figura 3.29: Exemplo de parâmetros para o VR Signal Expander

Na saída deste bloco deverá simplesmente acontecer uma troca de coordenadas dando origem a um vector de saída, $v_{out} = [1\ 3\ 2; 1\ 3\ 3; 1\ 3\ 4]$.

VR Placeholder

Muitas vezes ao utilizarmos o "VR Signal Expander" pretende-se usar apenas alguns dos valores do sinal de entrada para efectuar controlo de um objecto numa cena 3D. Perante esta situação a solução é usar o bloco "VR Placeholder" (figura 3.30). Este bloco permite enviar um valor não especificado para um bloco da VRT.

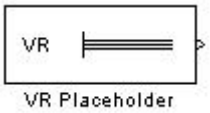

Figura 3.30: Bloco VR Placeholder

De forma a perceber o seu funcionamento, abordemos o exemplo mencionado anteriormente. Imaginemos agora o caso em que possuímos um vector de entrada,$v1 = [2\ 1\ 3; 3\ 1\ 3; 4\ 4\ 3]$, construído segundo o sistema de coordenadas do Simulink® e do qual pretendemos usar apenas a coordenada correspondente à latitude ou abcissa. Neste caso, ao colocarmos o vector v na entrada do bloco VR Signal Expander deveríamos realizar as configurações nas caixas de diálogo dos dois blocos como consta na figura 3.31.

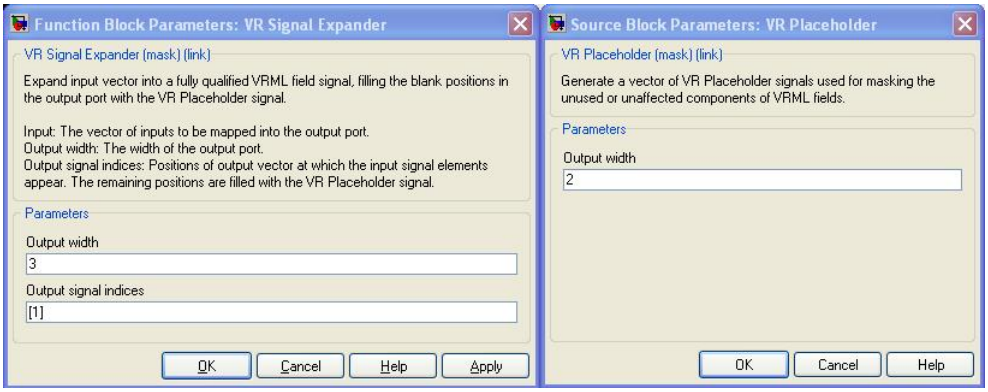

Figura 3.31: Configuração das caixas de diálogo dos blocos VR Signal Expander e VR Placeholder

Com as configurações feitas acima e tendo em conta o funcionamento de ambos os blocos envolvidos nesta operação, o vector de saída do "VR Signal Expander" será dado por $v0 = [2\ NaN\ NaN; 3\ NaN\ NaN; 4\ NaN\ NaN]$, onde NaN *(not a number)* é a representação do Matlab® para um valor não especificado. Ao nível da interacção deste vector com o "VR Sink", o que se verifica é que os valores numéricos são passados ao campo do nó que estamos a controlar e os valores não numéricos, identificados por NaN, mantêm os seus valores apropriados à cena virtual.

VR Text Output

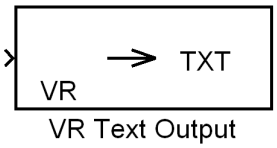

Figura 3.32: Bloco VR Text Output

O bloco "VR Text Output" (figura 3.32) tem como objectivo a impressão de texto directamente na cena 3D criada e a sua animação. A utilização deste tipo de blocos implica a configuração da caixa de diálogo como se apresenta na figura 3.33.

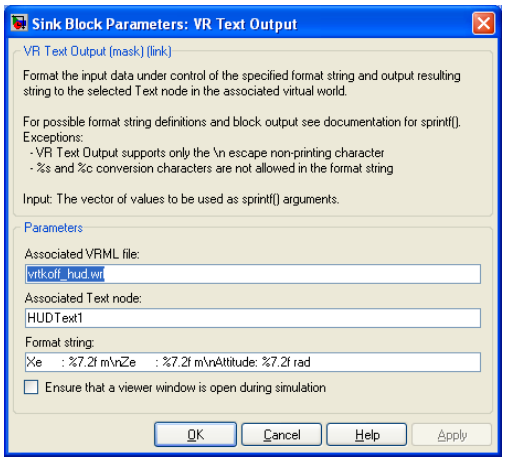

Figura 3.33: Configuração da caixa de diálogo do bloco VR Text Output

A figura 3.33 demonstra a configuração das propriedades do nó de texto do demo "vrtkoff_hud.wrl" fornecido pelo software Matlab® (executar na janela de comandos: vrdemos). Os parâmetros associados permitem definir qual o ficheiro que contém o mundo virtual ("Associated VRML file"), o nó do tipo texto ("Associated Text node") e ainda o formato de saída da *string* ("Format string").

Outros Blocos

A VRT disponibiliza outros blocos funcionais para além daqueles mencionados nos parágrafos anteriores. Esses blocos, no entanto, e tendo em conta o trabalho que se pretende desenvolver, não justificam uma análise tão exaustiva. Apesar deste facto, apresentam-se em baixo (figura 3.34) todos os blocos disponibilizados pela biblioteca de funcionalidades da VRT.

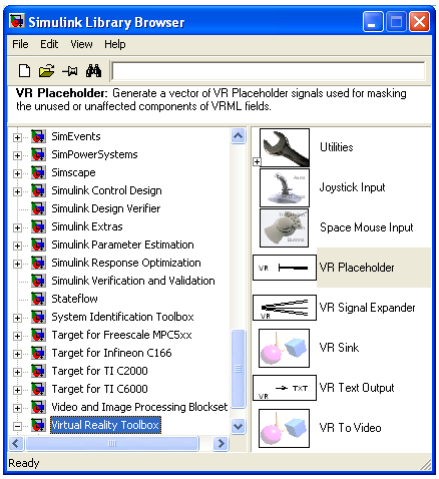

Figura 3.34: Biblioteca de blocos da VRT

Dentro da gama de funcionalidades apresentada na figura 3.34, há ainda algumas utilidades dentro do bloco seleccionado. Entre estas funcionalidades uma nota especial para os blocos "Rotation Matrix to VRML Rotation" e "Rotation Between 2 Vectors" (figura 3.35), os quais possibilitam o cálculo do ângulo de rotação e a respectiva conversão para o formato de entrada nos blocos da VRT.

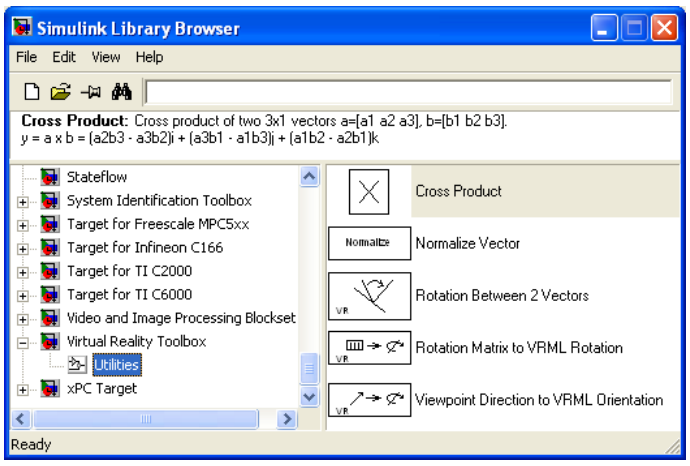

Figura 3.35: Utilidades da biblioteca da VRT

Capítulo 4

Casos de Estudo

No presente ponto apresentam-se três casos de estudo de sistemas de controlo. Estes foram analisados com base nos modelos matemáticos lineares que os caracterizam e foram implementados, posteriormente, ao nível de um ambiente virtual, recorrendo ao V-Realm Builder 2.0 e à VRT.

4.1 Controlo de Velocidade de um Veículo

4.1.1 Introdução

O primeiro caso de estudo visa projectar e controlar a velocidade de um automóvel num ambiente virtual, sendo este controlo realizado segundo o modelo matemático calculado e visualizado em tempo real no mundo virtual.

Para se obter o modelo matemático que representa a dinâmica de um veículo de massa (M) considerou-se uma força (F), a ser gerada pelo motor, e uma força de atrito contrária ao seu movimento (f). O esquema de forças representa-se na figura abaixo (ver figura 4.1). Neste caso assumiu-se como desprezável o atrito das rodas com a superfície de contacto. Por esse motivo, a força de atrito (f) depende unicamente da resistência que o ar oferece ao movimento do carro.

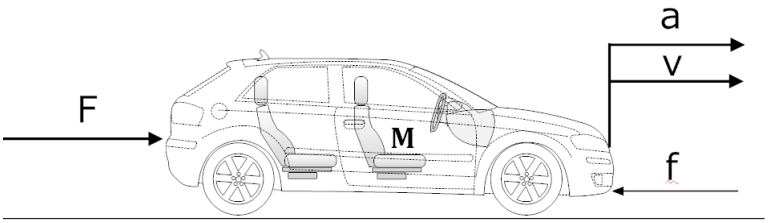

Figura 4.1: Esquema de forças da dinâmica de um veículo

Na figura 4.1 estão representadas não só as forças que actuam no automóvel mas também a velocidade (v) e aceleração (a).

Para a concretização da experiência foram usados os dados fornecidos em [10] para os valores de massa (M) e da força de atrito (f), bem como para o tempo de simulação (tsim). Estes dados são definidos a partir de um *script* em Matlab® (ver anexo 6.1) e exibidos no algoritmo 4.1

Algorithm 4.1 Definição dos dados do sistema automóvel

```
%Dados do Sistema
tsim=60;%tempo de simulação do modelo em segundos
M=700; %massa do veículo em Kg
f=50; %força de atrito em N.m
```

Para se projectar o controlador de velocidade para o veículo é necessário ter em conta as especificações de dinâmica que se pretendem atingir. O caso de estudo em análise tem como especificações

de regime estacionário e transitório os valores definidos em [10] e listados em baixo.

- No regime transitório devem ser cumpridos:

1. Sobre-elevação ("Overshoot") máximo de 10%;
2. Tempo de subida ("Rise time") $<$ 10 segundos;

- No regime estacionário devem ser atingidos:

1. Velocidade de referência de 10 m/s;
2. Erro em regime estacionário (ess) $<$ 2% da velocidade de referência;

4.1.2 Obtenção do Modelo Matemático Linear

O modelo matemático linear do veículo pode ser obtido a partir da Primeira Lei de Newton[1] (ou Princípio da Inércia). Desta forma chega-se à equação 4.1, a qual relaciona todas as forças actuantes no veículo:

$$M \cdot a(t) = F(t) - f \cdot v(t) \tag{4.1}$$

Como se pretende controlar a velocidade do automóvel temos que exprimir a equação acima em função da variável (v):

$$a(t) = \frac{dv(t)}{dt} = \dot{v} \tag{4.2}$$

A partir da substituição da equação 4.2 na equação 4.1 obtém -se a equação 4.3:

$$M \cdot \dot{v} = F - f \cdot v \tag{4.3}$$

De modo a facilitar o uso da informação sobre a dinâmica do sistema fornecida pela equação acima, utilizou-se a transformada de Laplace e consideraram-se nulas as condições iniciais para se obter a saída do sistema em função das outras variáveis em jogo. Assim, relacionou-se a velocidade (v) do veículo, a força (F) gerada pelo motor e a força de atrito (f) que se opõe ao movimento, obtendo-se a equação 4.4:

$$V(s) = \frac{F(s)}{M \cdot s} - \frac{V(s) \cdot f}{M \cdot s} \tag{4.4}$$

$$G(s) = \frac{V(s)}{F(s)} = \frac{1}{M \cdot s + f} \tag{4.5}$$

Os modelos matemáticos obtidos nas equações 4.4 e 4.5 podem agora ser implementados virtualmente, recorrendo à ferramenta Simulink® e a vários dos seus blocos ou simplesmente usando o bloco "Transfer Fcn". Tendo em conta uma abordagem do tipo *top-down* e usando a representação matemática obtida na equação 4.4, criou-se inicialmente o subsistema "Sistema do Carro" (figura 4.2).

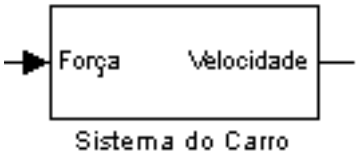

Figura 4.2: Bloco representativo do modelo virtual do automóvel

[1] A Primeira Lei de Newton enuncia que :"A soma algébrica das forças que actuam sobre um corpo rígido numa dada direcção é igual ao produto entre a massa desse corpo e a sua aceleração na mesma direcção."[24]

Seguidamente, interligaram-se os blocos do Simulink dentro deste subsistema e respeitando a equação 4.4 obtida anteriormente. A representação do sistema apresenta-se na figura 4.3.

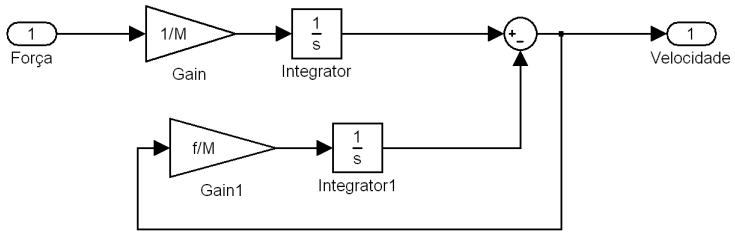

Figura 4.3: Representação do sistema de um veículo em Simulink

4.1.3 Elaboração do Mundo Virtual

Para a elaboração do mundo virtual recorreu-se ao editor V-Realm Builder 2.0 fornecido pelo Matlab®. A partir desta ferramenta construiu-se um ambiente virtual em 3D que vai ser controlado através dos dados gerados pelo modelo matemático da dinâmica de um veículo.

Sendo o mundo virtual uma peça de visualização da acção do controlador na evolução da dinâmica do carro, pretende-se um ambiente em que seja possível observar o automóvel na sua posição inicial e acompanhar o seu estado de movimento ao longo do tempo de simulação. O resultado obtido encontra-se na figura 4.4:

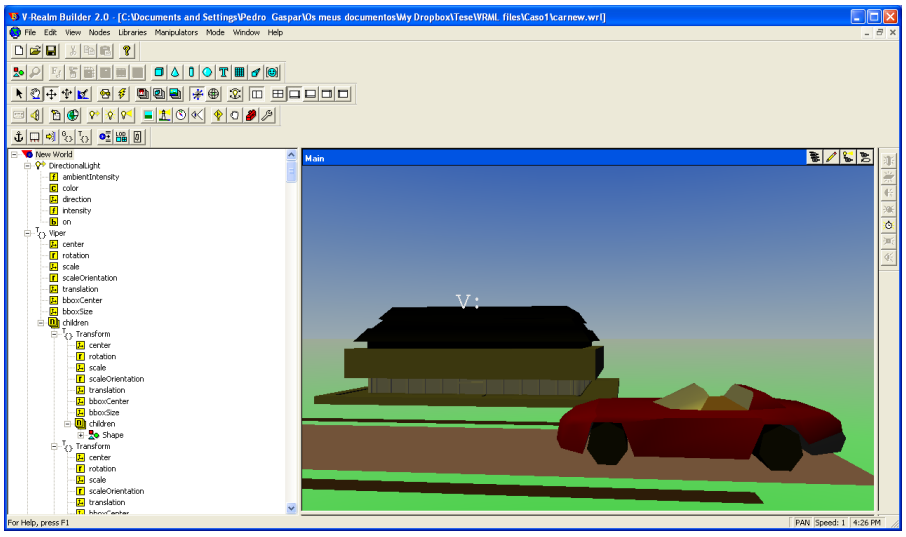

Figura 4.4: Ficheiro .wrl de um mundo virtual com um automóvel

Na criação deste mundo virtual foi necessário ter em conta que se pretendia controlar algumas características de alguns objectos presentes na cena. Devido a esse facto atribuiram-se nomes aos nós representantes dos objectos que se pretendem controlar. Neste caso específico adicionou-se um painel de texto que acompanha também o movimento do carro e que informa o utilizador do valor da velocidade instantânea.

4.1.4 Integração do Mundo Virtual com a Virtual Reality Toolbox

Depois de se ter obtido o modelo linear que representa a dinâmica do carro e se ter criado uma cena 3D que se adeque ao processo que pretendemos visualizar, é possível fazer interagir o primeiro com o último. Para esse fim usou-se a VRT e os seus blocos, obtendo-se o diagrama de blocos na figura 4.5, para visualização da animação no ambiente virtual 3D .

Nesta situação específica tentou-se colocar em movimento o carro presente na cena 3D apresentada em 4.1.3. Com a função matemática que se insere no bloco "Sistema do Carro" obtemos, na saída do mesmo, a velocidade que este vai atingindo ao longo do tempo. Como já foi referido, a velocidade não é um atributo compatível com os blocos da VRT, sendo por isso necessária a introdução de um bloco "Integrator" no seguimento do sinal que sai do bloco "Sistema do Carro". Através deste integrador obtém-se a posição do carro durante o período de simulação.

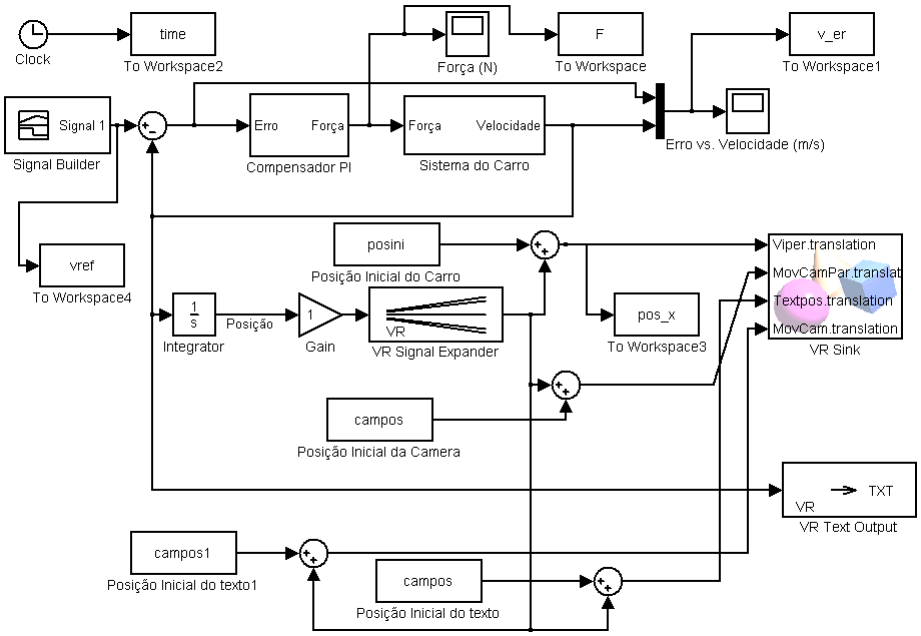

Figura 4.5: Diagrama de blocos da integração da cena 3D com o modelo matemático de um veículo em Simulink

Na integração destes dois modelos é necessário levar em consideração algumas das características abordadas sobre os ambientes descritos em linguagem VRML. Assim, descrevem-se nos parágrafos seguintes os passos tomados de forma a obter um sistema funcional.

Antes de o sinal "Posição" estar totalmente compatível com o movimento que se pretende incutir ao carro, necessitamos de converter este sinal num vector do tipo $[x\ y\ z]$. A conversão é feita pela configuração do bloco "VR Signal Expander" como demonstra a figura 4.6.

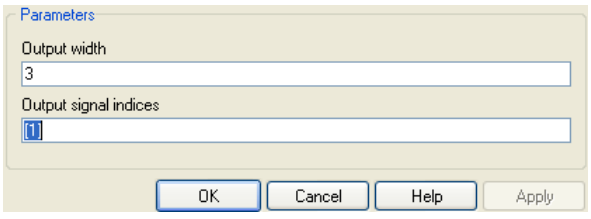

Figura 4.6: Conversão do sinal "Posição" num vector 1x3 do tipo $[x\ y\ z]$.

Neste momento o sinal já está devidamente acondicionado para ser passado ao bloco "VR Sink". Porém, devido ao facto de a posição inicial do automóvel não ser a origem, ou seja, não estar centrado na posição [0 0 0], é necessário efectuar uma translação a partir da posição inicial em que se encontra na cena 3D. A posição inicial de qualquer objecto pode ser obtida a partir do campo "*translation*" do nó que o representa, ou através da execução de comandos que acessam este tipo de nós nos ficheiros *.wrl*, como é demonstrado no algoritmo 4.2:

Algorithm 4.2 Acesso aos nós VRML

```
wrl=vrworld('carnew.wrl'); %inicialização do mundo virtual
x=nodes(wrl,'-full'); %carregamento dos nós do mundo virtual numa variável
posini=wrl.Viper.translation; %leitura da posição inicial do objecto
```

Por fim, falta apenas informar o bloco "VR Sink" qual o nó e o respectivo campo que pretendemos preencher com a informação gerada pelo diagrama de blocos. Essa acção é feita através da caixa de diálogo do bloco "VR Sink", na qual necessitamos de preencher de acordo com o que foi já mencionado em 3.3.7e como mostra a figura 4.7.

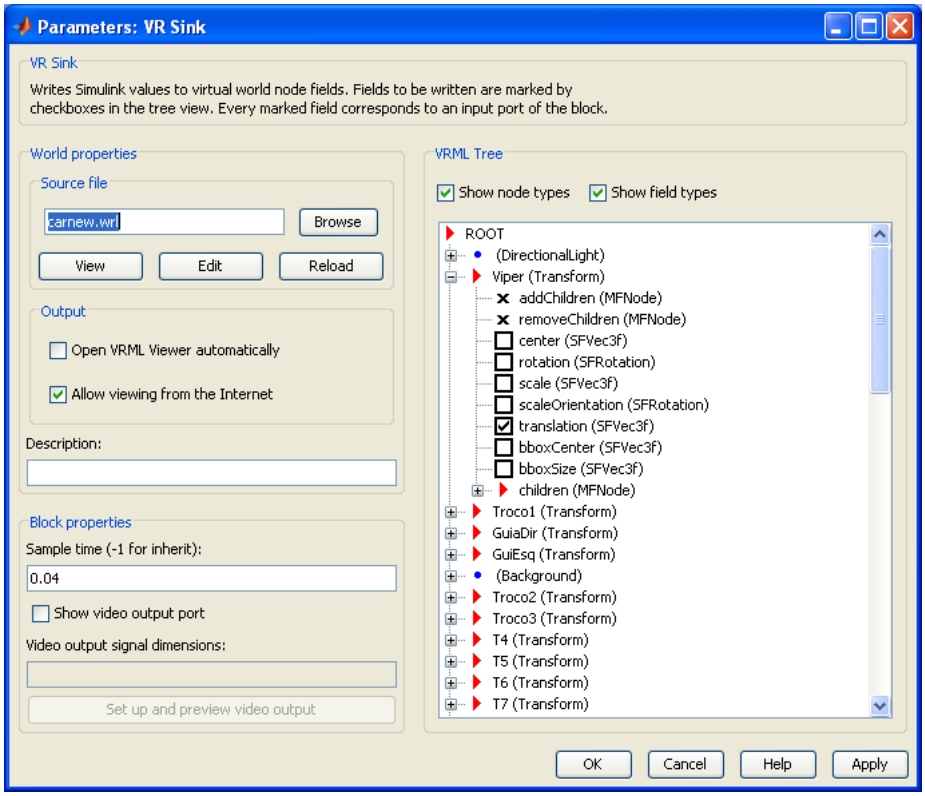

Figura 4.7: Preenchimento da caixa de diálogo do bloco "VR Sink" do modelo virtual de um controlador de velocidade para um veículo

Com esta configuração é já possível verificar o funcionamento integrado do modelo em Simulink® conjuntamente com os blocos funcionais da VRT e ainda com o mundo virtual que se construiu. Como sinal de referência para o teste do modelo implementado usou-se o bloco "Signal Builder" com um degrau inicial de $10\,m/s$ e uma alteração da referência para $20\,m/s$ em $t = 20$ segundos. De modo a se poder comparar o valor real da velocidade atingida com o valor de referência definido usou-se o bloco "To Workspace4".

4.1.5 Controlo de Velocidade de um Veículo com Compensador PI (Proporcional - Integrador)

Utilizando o modelo linear que representa a dinâmica do sistema obtido em 4.1.2, é agora necessário calcular o modelo dinâmico do compensador que cumpre com as especificações citadas.

Os compensadores usados em sistemas de controlo caracterizam-se por trabalharem directamente com o sinal de erro. O processamento que é feito deste sinal tem como objectivo a produção de um novo sinal de saída do bloco compensador de forma a que o sistema se comporte dentro das especificações exigidas e de acordo com as suas limitações físicas. Tendo em conta que a função de tranferência

do sistema em malha aberta apresenta apenas um pólo deslocado da origem (sistema de 1ª ordem) testou-se o modelo com um compensador do tipo P (proporcional):

$$c_P(t) = K_p e(t) \tag{4.6}$$

Onde $e(t)$ é o sinal de erro e é definido por:

$$e(t) = vref - v(t) \tag{4.7}$$

Calculou-se em seguida a função de transferência (FT) do compensador no domínio de Laplace, cuja equação é dada por:

$$K_p(s) = \frac{C_P(s)}{E(s)} = K_p \tag{4.8}$$

Considerando uma topologia de realimentação negativa e a equação 4.5, obtida anteriormente, chega-se à FT do sistema em malha fechada ($CL(s)$) (equação 4.9). A partir desta equação, das especificações exigidas para o comportamento do sistema e das técnicas de controlo clássico estudadas durante a componente prática da cadeira de Sistemas e Controlo 2, obtiveram-se os resultados seguintes para o projecto do compensador:

$$CL(s) = \frac{K_p}{Ms + f + K_p} \tag{4.9}$$

$$ess = \lim_{s \to 0} s \cdot E(s) = \lim_{s \to 0} s \cdot \frac{Vref(s)}{1 + K_p(s) \cdot G(s)} \tag{4.10}$$

Substituindo $G(s)$ e $K_p(s)$ na equação 4.10, assim como o valor especificado para o erro em regime estacionário, e considerando como sinal de referência ($Vref(s)$) o sinal gerado pelo bloco "Signal Builder" (que inicialmente apresenta um degrau de amplitude $10\,m/s$), obteve-se um valor de $K_p = 2450$.

Seguidamente, através da execução do código presente no anexo6.1, resultou a figura 4.9.

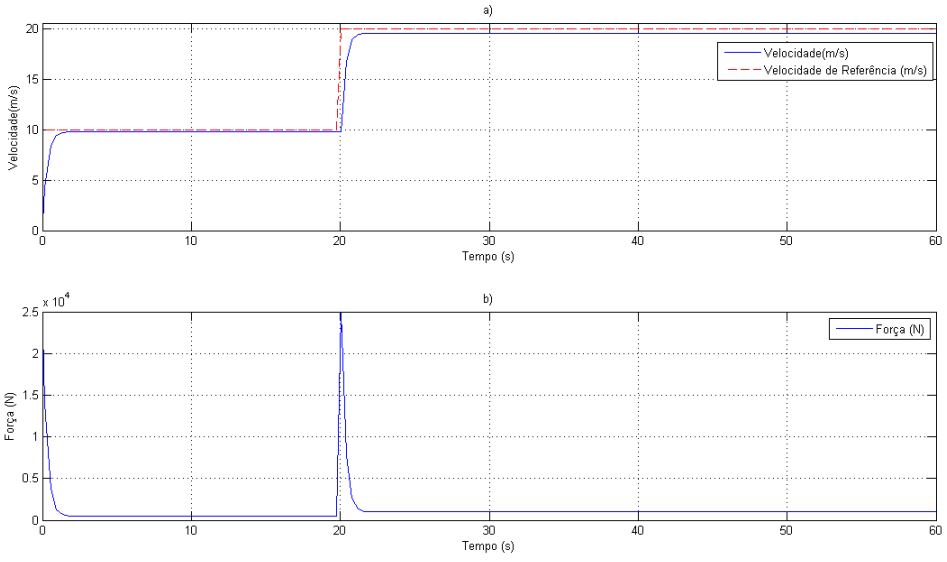

Figura 4.8: Comportamento do sistema ao longo do tempo usando um compensador P

Dos resultados apresentados conclui-se que as especificações definidas para o regime transitório são cumpridas, bem como as características da resposta em regime estacionário. Ou seja, de acordo com o gráfico a) é visível que não existe sobre-elevação, o tempo de subida é respeitado e que o erro em regime estacionário é cumprido. Conclui-se assim que o compensador proporcional seria suficiente

para controlar a força gerada pelo motor de modo a se cumprir para com a dinâmica pretendida. No entanto, aumentando o valor da constante de proporcionalidade do controlador atingem-se limites físicos de realização da compensação, pois a força à qual é submetido o motor do carro ultrapassa a resistência dos materiais que o constituem. Por este motivo há que desenvolver um compensador capaz de respeitar a dinâmica pretendida, mas que vá de encontro aos valores reais de força a que se pode subemeter o motor do carro.

Das considerações anteriores, e tendo em conta a tabela 4.1, verifica-se que o mais apropriado para compensar um sistema de primeira ordem com um pólo real, é um compensador PI (Proporcional-Integrador).

	Regime transitório			Regime estacionário
Coeficiente do Compensador	tempo de subida (t_r)	sobreelevação (PO %)	tempo de estabelecimento (t_s)	erro em regime estacionário (ess)
K_p	Diminui	Aumenta	Pouco efeito	Diminui mas não anula (I)
K_i	Diminui	Aumenta	Aumenta	Anula ess (I,II)
K_d	Pouco efeito	Diminui (III)	Diminui (III)	Pouco efeito

Tabela 4.1: Comportamento dos regimes transitório e estacionário do sistema com o aumento dos coeficientes do compensador

I - Verifica-se apenas para sistemas sem pólos na origem;

II - Principal vantagem do aumento do coeficiente K_i;

III - Principal vantagem do aumento do coeficiente K_d;

Partindo das especificações de dinâmica mencionadas pretende-se projectar um compensador PI (Proporcional-Integrador), cuja equação diferencial é definida por:

$$c_{PI}(t) = K_p \cdot e(t) + K_i \cdot \int e(t)dt \tag{4.11}$$

Onde $c(t)$ representa o sinal de saída do compensador, K_p é a constante de proporcionalidade, K_i a constante de ganho integrador e $e(t)$ é o sinal de erro do sistema (ver equação 4.7). Aplicando a transformada de Laplace à equação 4.11, obtém-se a seguinte FT para o compensador PI:

$$K_{PI}(s) = \frac{C_{PI}(s)}{E(s)} = K_p + \frac{K_i}{s} \tag{4.12}$$

No Simulink® implementou-se o compensador da seguinte forma:

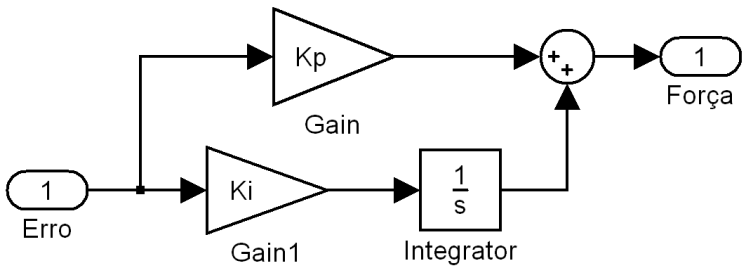

Figura 4.9: Realização do compensador PI com blocos Simulink®

Considerando a FT obtida na equação 4.5, representativa da dinâmica do carro, e o esquema de controlo evidenciado pela figura 4.5, calculou-se a nova FT $CL(s)$ do sistema em malha fechada. O

projecto do compensador depende directamente das características que se pretende que o sistema apresente nos regimes transitório e estacionário. Assim, calculou-se $CL(s)$ na forma[2] $\frac{A\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$, obtendo-se a equação 4.13. Deste modo, obtem-se facilmente o valor das constantes proporcional e constante de ganho integral.

$$CL(s) = \frac{(K_p \cdot s + K_i)/M}{s^2 + \left(\frac{f+K_p}{M}\right) s + \frac{K_i}{M}} \tag{4.13}$$

Relacionando os denominadores das duas equações obtêm-se as igualdades $2\xi\omega_n = \left(\frac{f+K_p}{M}\right)$ e $\omega_n^2 = \frac{K_i}{M}$. Das igualdades referidas, das especificações de dinâmica[3] definidas para o controlo de velocidade de um veículo, obtiveram-se respectivamente $K_p \simeq 281.4$ e $K_i \simeq 57.5$.

A partir dos valores de base calculados ajustaram-se os valores dos ganhos para $K_p \simeq 207.8$e $K_i \simeq 25.4$. e executou-se o modelo apresentado anteriormente (figura 4.5) através do código no anexo 6.1. Os resultados obtidos apresentam-se na figura 4.10.

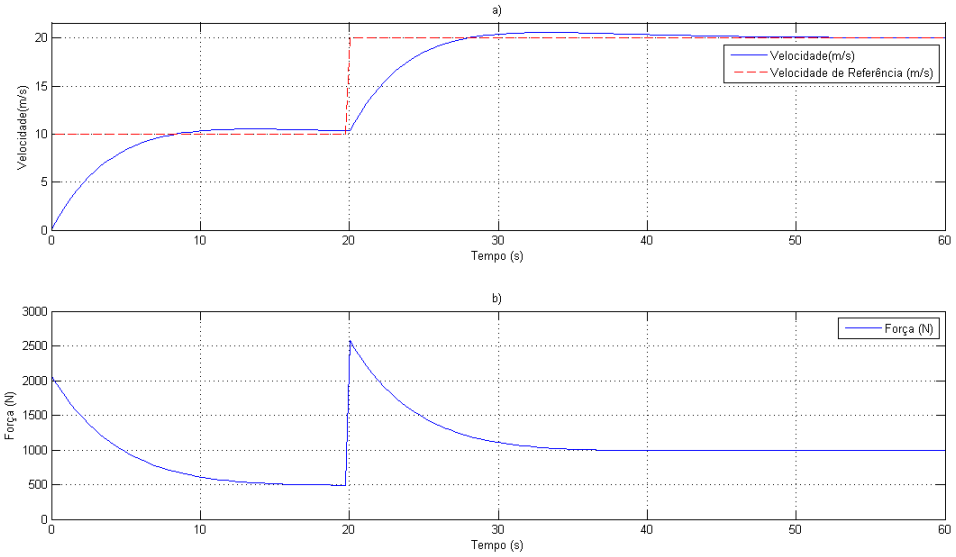

Figura 4.10: Resposta do sistema no tempo usando um controlador PI

Como é perceptível, os resultados produzidos pelo teste deste sistema com o compensador PI projectado demonstram (ver figura 4.10) que é possível realizar controlo de velocidade do veículo cumprindo todas as especificações de dinâmica impostas e através de menores valores de força aplicados ao carro. As especificações de tempo de subida (tr) e erro em regime estacionário (ess) foram atingidas com sucesso devido ao comportamento integrador que o compensador implementa, assim como o factor de sobre-elevação que não ultrapassa os 10% definidos. De notar que no gráfico a) apenas na resposta à alteração de referência (20 m/s em $t = 20$ segundos) se anula o erro em regime estacionário, dado que para o sinal de referência de 10 m/s o tempo necessário à estabilização não é suficiente.

Em termos físicos os valores obtidos para as constantes proporcional e integral resultam numa diminuição da força que é requerida ao motor do veículo, traduzindo-se este facto pela curva no gráfico b).

Relativamente ao funcionamento do mundo virtual criado para esta situação, visualiza-se na figura 4.11 o estado da execução do ambiente 3D para $t = 10$ segundos. Neste caso observa-se que o valor da velocidade se encontra na vizinhança definida pelo erro em regime estacionário.

[2] Forma genérica da função de transferência de um sistema de 2º ordem sem zeros

[3] As características do sistema em regime transitório permitem calcular o valor dos parâmetros ξ (coeficiente de amortecimento) e ω_n(frequência natural) a partir das equações: $\xi = \sqrt{\frac{(\ln(PO))^2}{(\ln(PO))^2+\pi^2}}$e $\omega_n \approx \frac{0.8+2.5\xi}{t_r}$[10].

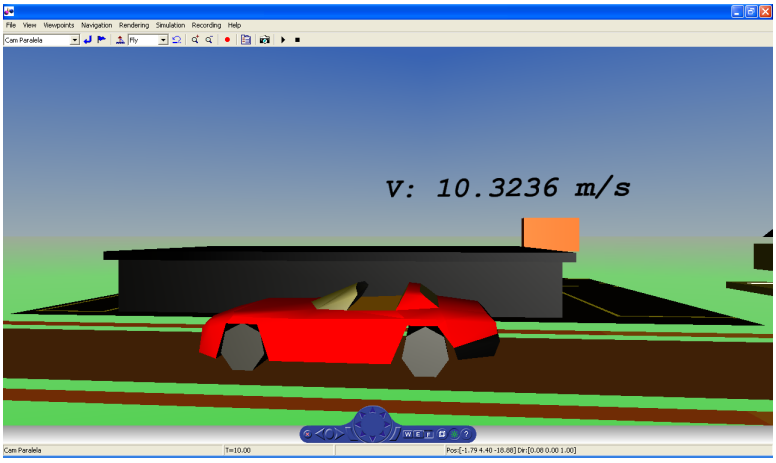

Figura 4.11: Resposta no mundo virtual com compensador PI, para $t = 10$ segundos

4.1.6 Controlo de Velocidade de um Veículo com Compensador Atraso

O comportamento de um compensador atraso aproxima-se daquele que é implementado pelo compensador projectado anteriormene, ou seja, o controlador PI. A FT que define a dinâmica de um compensador atraso é dada por:

$$\frac{C_{at}(s)}{E(s)} = K_{at} \cdot \frac{s+a}{s+\lambda \cdot a} \begin{cases} a > 0 \\ 0 < \lambda < 1 \end{cases} \tag{4.14}$$

A FT que descreve o compensador atraso depende dos parâmetros a e λ e do ganho K_{at}. Estes parâmetros são definidos de modo a se obter um comportamento mais lento do sistema. Assim, definiu-se empiricamente $\lambda = 0.1$ e a como o módulo do pólo mais dominante do sistema sem compensação ($a = 0.07$). O ganho K_{at} funciona como um parâmetro de ajuste de modo a que se cumpram as especificações de dinâmica.

Calculando a FT do sistema em malha fechada a partir das equações 4.14 e da equação que modela a dinâmica do carro (equação 4.5) obteve-se a relação na equação 4.15.

$$G_{cl}(s) = \frac{V(s)}{F(s)} = \frac{\frac{K_{at} \cdot (s+a)}{M}}{s^2 + s(\frac{f + a \cdot \lambda \cdot M + K_{at}}{M}) + \frac{\lambda \cdot a \cdot f + a \cdot K_{at}}{M}} \tag{4.15}$$

A partir do código no anexo 6.1 e da fórmula genérica para um sistema de 2ª ordem sem zeros, calculou-se o valor de base para o ganho K_{at}, o qual foi posteriormente ajustado iterativamente.

Da execução do modelo na figura 4.13 verificou-se que os valores de força gerados eram da ordem dos obtidos com o controlador tipo PI. Assim, projectou-se $K_{at} = 213.6$ executou-se o modelo na figura 4.13 obtendo-se os resultados na figura 4.12.

Comparando a realização do controlo do sistema com um compensador PI e o compensador atraso projectado (figura 4.14), conclui-se que o comportamento do automóvel é aproximadamente igual, usando os dois tipos de controladores, denotando-se um ligeiro atraso de fase na resposta do controlador atraso. Fisicamente denotam-se algumas diferenças que se podem revelar importantes para a escolha do controlador a usar. Relativamente às especificações de dinâmica em regime transitório conclui-se que ambos os controladores projectados atingem as características desejadas. O tempo de subida (tr) é inferior no caso do PI, enquanto o factor de sobre-elevação é mais elevado. Destas características observa-se em b) que a resposta mais lenta proporcionada pelo compensador atraso requer valores de força ao motor semelhantes aos obtidos com o controlador PI.

Desta análise pode ainda afirmar-se que com controlador atraso implementado, o erro não é nulo no regime estacionário, ao contrário do que se verifica com o controlador PI.

Em suma, o compensador PI revela-se a melhor escolha para o controlo da velocidade do veículo em questão. Para além de cumprir as especificações impostas (ambos os controladores cumprem), necessita de um menor ganho proporcional (K_p do PI é menor que K_{at} do atraso), os valores de força,

apesar de muito semelhantes, são menores no compensador Proporcional-Integrador e apesar de o PI apresentar sobre-elevação, responde mais rapidamente ao sinal de referência.

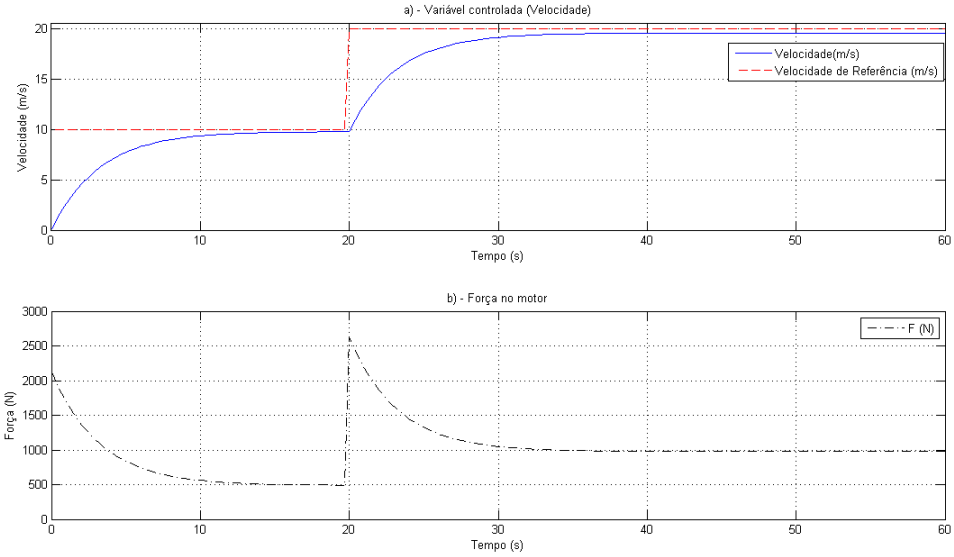

Figura 4.12: Resposta no tempo do sistema com compensador atraso

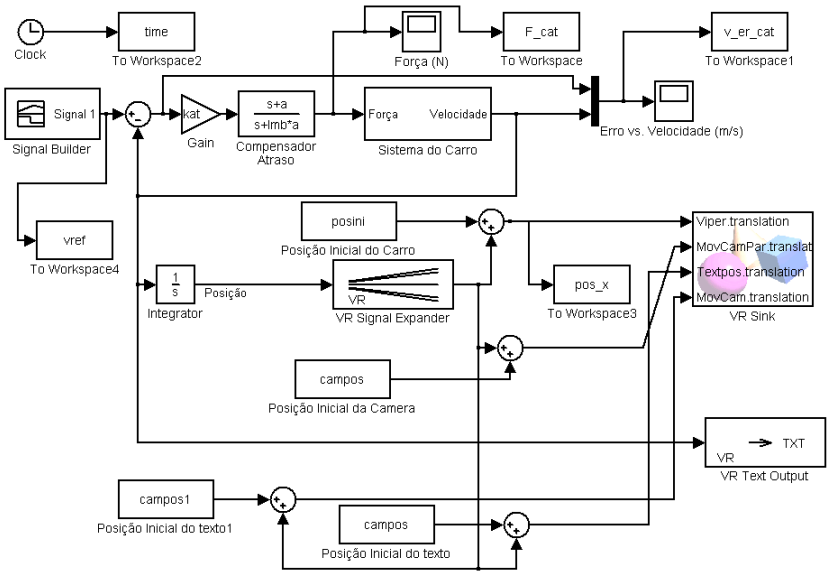

Figura 4.13: Integração do modelo 3D com Simulink® e com a VRT.

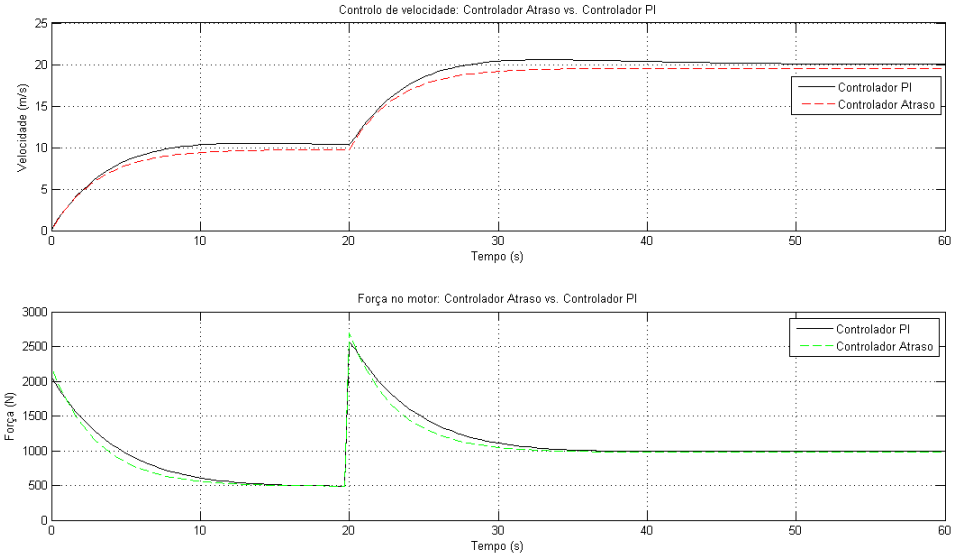

Figura 4.14: Comparação da performance dos 2 controladores projectados.

4.2 Controlo de Velocidade Angular de Saída de um Motor DC

4.2.1 Introdução

Neste caso de estudo pretende-se efectuar o controlo de velocidade angular na saída de um motor DC e visualizar em ambiente virtual o comportamento integrado deste sistema com um sistema de roldanas. Estas, consequentemente, accionam um tapete rolante de uma linha de montagem.

O esquema do sistema que se pretende implementar encontra-se na figura 4.15.

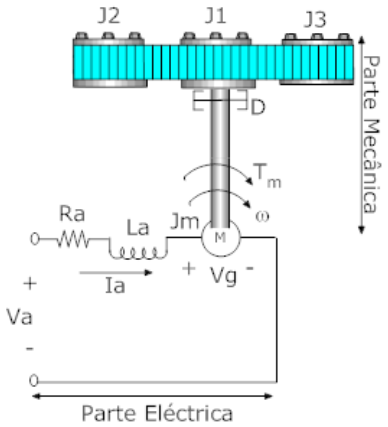

Figura 4.15: Esquema de funcionamento de uma linha de montagem controlado por um motor eléctrico DC

De acordo com a imagem apresentada, o sistema apresenta vários parâmetros constantes, descritos seguidamente:

- R_a- Resistência da armadura do motor (Ω)
- L_a- Inductância da armadura do motor (H)

- Vg - Tensão (V) devido à força contra electro-motriz ($Vg = K_g \cdot \omega$, onde K_g é a constante de contra-emf ($V \cdot s/rad$))
- Va - Tensão (V) aplicada aos terminais do motor e à sua armadura
- T_m- Binário ($N \cdot m$) produzido pelo motor ($T_m = K_m \cdot I_a$, onde K_m representa a constante de torque do motor ($N \cdot m/A$))
- D - Força de atrito provocada pela roldana relativamente ao eixo de saída ($N \cdot m \cdot s$)
- $J1$, $J2$ e $J3$ - Momento de inércia das roldanas ($Kg \cdot m^2$)
- Jm - Momento de inércia do motor relativamente ao eixo do motor ($Kg \cdot m^2$)

O projecto do controlador baseia-se no cumprimento de determinadas especificações dinâmicas por parte do mecanismo implementado. Desse modo, o conjunto motor DC - linha de montagem deve reagir ao sinal de referência respeitando as seguintes características:

- Regime transitório:

1. Sobre-elevação ("Overshoot") máxima de 10%;

- Regime estacionário:

1. Tempo de estabelecimento ("Settling Time") inferior a 10 segundos;
2. Erro em regime estacionário (ess) máximo de 0.02 relativamente à velocidade angular de referência;

4.2.2 Cálculo do Modelo Matemático Linear

Para a obtenção do modelo matemático que rege o sistema caracterizado no ponto anterior, foram tidos em consideração os seguintes aspectos: momento de inércia de $J2$ e $J3$ desprezável, o binário de sentido contrário a T_m e provocado pela carga $J1$ também negligenciável, assim como a massa dos objectos que circulam na linha de montagem. No cálculo da FT no domínio de Laplace consideraram-se condições iniciais nulas para a parte eléctrica e mecânica do sistema. Ou seja, inicialmente não existe qualquer energia armazenada na componente eléctrica e a parte mecânica encontra-se em repouso.

A equação que rege o funcionamento do circuito que implementa o motor DC baseia-se na equação das malhas da parte eléctrica na figura 4.15 e é dada pela equação 4.16.

$$-Va(t) + Ia(t) \cdot Ra + La \cdot \frac{dIa(t)}{dt} + Vg = 0 \tag{4.16}$$

Relativamente ao sistema mecânico que está acoplado ao motor DC (parte mecânica na figura 4.15) podemos definir a equação 4.17 pela 1ª Lei de Newton.

$$\alpha(t) \cdot (Jm + J1) = T_m(t) - D \cdot \omega(t) \tag{4.17}$$

Onde $\alpha(t) = \frac{d\omega(t)}{dt} = \dot{\omega}$.

Substituindo as relações $Vg = K_g \cdot \omega$ e $T_m = K_m \cdot I_a$, respectivamente nas equações 4.16 e 4.17, resolvendo a primeira em ordem a Ia e substituindo o resultado na segunda obtém-se a equação que exprime a velocidade angular $\omega(t)$ em função da tensão $Va(t)$. Aplicando a transformada de Laplace, chega-se à FT do sistema em malha aberta $OL(s)$ (equação 4.18).

$$OL(s) = \frac{W(s)}{Va(s)} = \frac{\frac{K_m}{La \cdot (Jm+J1)}}{s^2 + s\frac{(Jm+J1)\cdot Ra + D\cdot La}{(Jm+J1)\cdot La} + \frac{D\cdot Ra + K_m . K_g}{(Jm+J1)\cdot La}} \tag{4.18}$$

Considerando o sistema de 2ª ordem obtido na forma $\frac{Y(s)}{X(s)} = \frac{b}{a2\cdot s^2 + a1\cdot s + a0}$, criou-se o algoritmo 4.3 seguinte (presente no anexo 6.2), cujos valores das constantes definidas para o sistema, foram retiradas de [30].

Algorithm 4.3 Definição de constantes e coeficientes para o modelo de um motor DC.

```
%Dados Motor DC
J1=0.01; %Inércia das "Roldana 1" (Kg.m^2)
Jm=0.01; %Inércia do motor (kg.m^2)
D=0.1; % Força de atrito no eixo de saída (N.m.s)
La=0.5; %Indutância da Armadura (H)
Ra=1; % Resistência da Armadura (ohm)
Km=0.01; % Constante do Binário produzido pelo motor
Kg=0.01; %Constante do cálculo da f.e.m.
%%%%%Valores do ambiente virtual
myworld=vrworld('DCmotor.wrl');
x=nodes(myworld,'-full');
r=(myworld.Rold1Rim.scale(1,1))/2; %Raio das "Roldanas"
campos=myworld.Camera_line.translation;
posini=myworld.Missile.translation;
%Coeficientes do sistema de 2ª ordem
a2=1;
a1=((J1+Jm)*Ra+D*La)/((J1+Jm)*La);
a0=(D*Ra+Km*Kg)/((J1+Jm)*La);
b=Km/((J1+Jm)*La);
```

Em Simulink® implementou-se o sistema na figura 4.16,

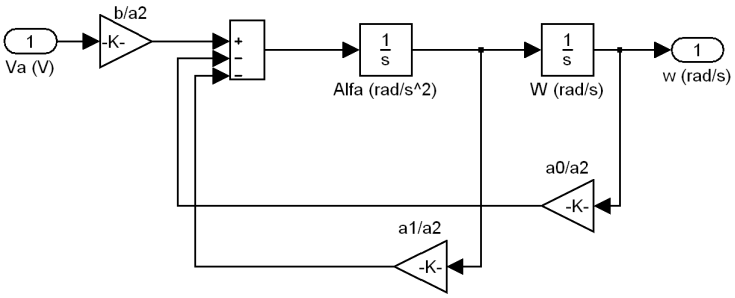

Figura 4.16: Modelo matemático linear do motor DC implementado em Simulink®.

E agruparam-se os blocos no subsistema "motorDC_roldana model" representado pela ilustração 4.17.

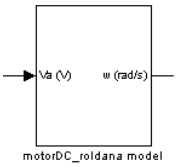

Figura 4.17: Subsistema do modelo matemático do motor DC.

4.2.3 Implementação do Ambiente Virtual

Atendendo ao esquema do sistema apresentado na figura 4.15 criou-se um ambiente 3D que permite a visualização do funcionamento do sistema de forma animada. O mundo virtual concebido apresenta-se na figura 4.18. Neste ambiente criaram-se também diferentes pontos de vista da cena por forma a observar-se o comportamento do sistema, quer ao nível da translação dos objectos na linha de montagem, quer do movimento de rotação das roldanas gerado pelo motor DC.

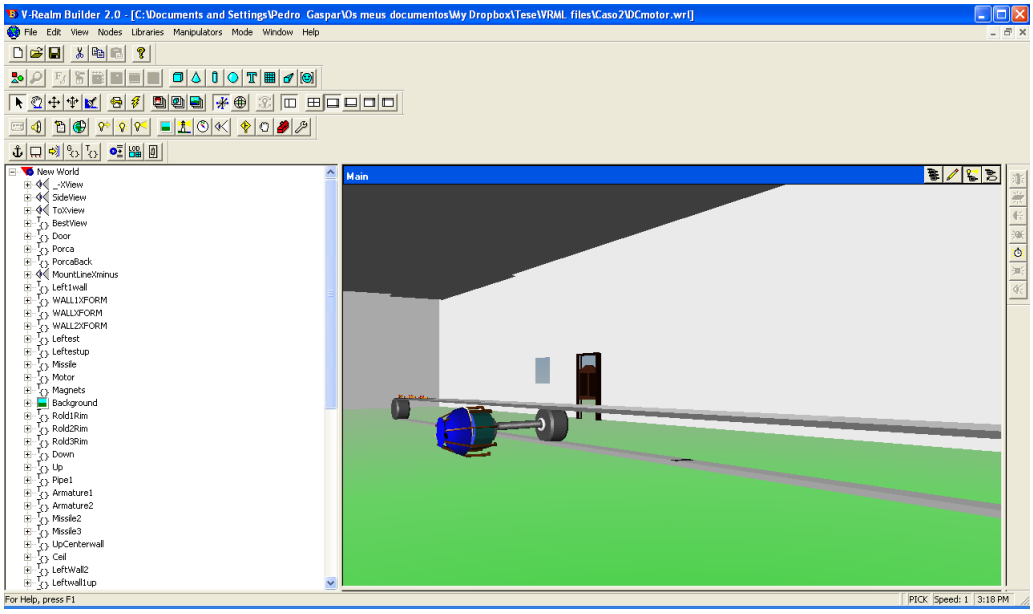

Figura 4.18: Ambiente virtual arquitectado para implementação de uma linha de montagem controlada por um motor DC.

Neste segundo caso de estudo pretende-se controlar a velocidade angular de saída de um motor DC. No entanto, este sistema electro-mecânico encontra-se acoplado a uma linha de montagem que transporta objectos num movimento de translação rectilíneo. Este facto exige que no ambiente 3D construído seja possível visualizar o comportamento dinâmico das principais peças do sistema. Para se poder controlar estas peças a partir do modelo matemático linear realizado em Simulink®, nomearam-se os nós dos objectos cujos campos se pretendem modificar.

4.2.4 Integração do Ambiente Virtual com o Simulink através da VRT

Neste momento é possível realizar a integração do ambiente virtual com o modelo implementado em Simulink® através dos blocos da VRT. O acoplamento das duas ferramentas é realizado como demonstra o diagrama de blocos na figura 4.19.

O esquema observado na figura foi criado, tal como para o caso de estudo anterior, de acordo com as imposições da linguagem VRML.

O bloco denominado por "motorDC_roldana model", descrito no ponto 4.2.2, tem como sinal de saída a velocidade angular gerada pelo motor DC. Para controlar a rotação de um objecto num ambiente virtual através do bloco "VR Sink", a velocidade angular não é a variável que se deve passar ao campo "rotation" do objecto. A rotação de um objecto é dada por um vector do tipo $[x\ y\ z\ \theta]$, o qual define em que eixo(s) se pretende efectuar a rotação e a posição angular. A posição angular do motor DC obtem-se a partir da velocidade angular, usando um bloco integrador na saída do bloco que representa o sistema motor-roldanas. O sinal resultante é depois multiplexado com o vector tridimensional $[0\ 0\ -1]$ (bloco "Axes") que define o sentido (horário) e eixo de rotação do objecto (eixo Oz).

A translação dos objectos que circulam na linha de montagem é obtida a partir da conversão da posição angular na posição linear (x). Este cálculo é feito com base na expressão 4.19.

$$x = \theta \cdot r \tag{4.19}$$

Onde θ é a posição angular e r é o raio da roldana. No diagrama de blocos este passo é executado através do produto do sinal de saída do integrador e da constante no bloco "Raio Roldana". Para que o movimento de translação seja executado ao longo de um determinado eixo, o resultado do produto anterior tem que ser colocado sob a forma de um vector do tipo $[x\ y\ z]$, sendo este obtido através do bloco "VR Signal Expander" com a configuração na figura 4.20. O sinal de saída do bloco "VR Signal Expander" necessita ainda de ser adicionado da posição original dos objectos para que

a translação ocorra a partir dessa posição. A posição inicial dos objectos foi obtida pela leitura do campo "Translation" do nó "Missile" na aplicação V-Realm Builder 2.0.

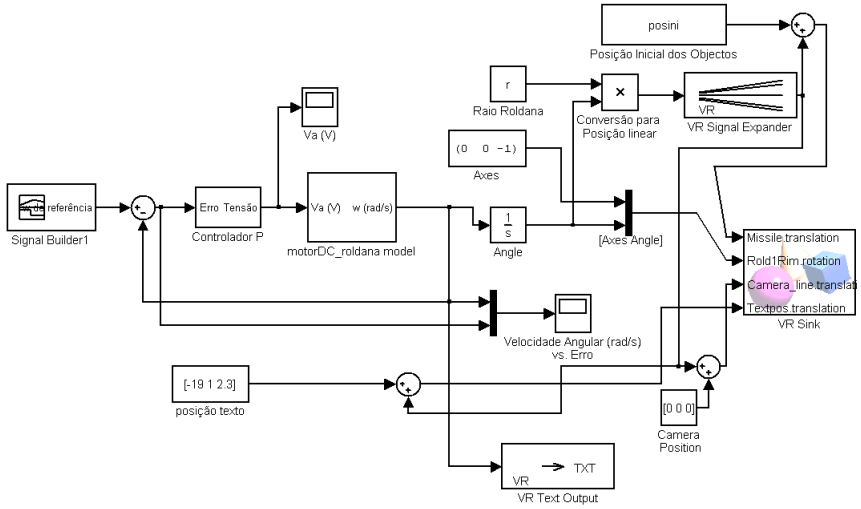

Figura 4.19: Esquema de integração em Simulink® do mundo virtual 3D com a os blocos da VRT.

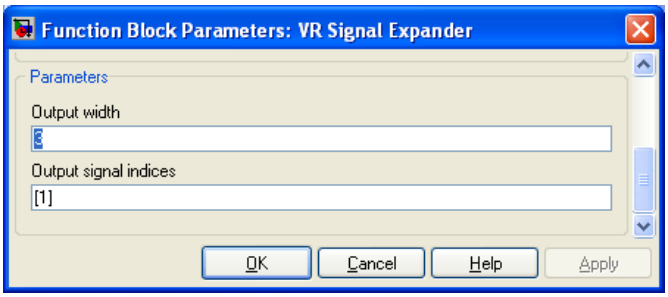

Figura 4.20: Configuração do bloco VR Signal Expander: translação dos objectos.

A cooperação deste modelo em Simulink® com o mundo virtual arquitectado ainda não está completa. Para que o ambiente 3D seja animado com os sinais de controlo gerados pelo modelo, é necessário efectuar a configuração de parâmetros do bloco "VR Sink" (ver figura 4.21).

De notar que apenas foram assinalados dois campos na configuração do bloco "VR Sink", o campo "Missile.translation" e o campo "Rold1Rim.rotation", quando se pretende controlar o movimento de rotação de quatro objectos (eixo do motor e as três roldanas) e o movimento de translação de outros três objectos (três mísseis).

Como o movimento que pretendemos controlar é sincronizado, utilizou-se neste caso específico a funcionalidade "Use", referida no ponto 3.3.5. Com esta função criaram-se duas roldanas a partir da primeira (nó "Rold1Rim") e dois míssseis a partir de um criado primeiramente (nó "Missile").

Para que haja uma maior interactividade com o utilizador procedeu-se à implementação de um painel de texto onde é impresso o valor da velocidade angular. Este objecto da cena 3D, cujo o nó foi denominado de "Textpos", é controlado a partir do modelo em Simulink® e do bloco "VR Text Output". A configuração do bloco "VR Text Output" e o nó criado para o efeito descrito está presente na figura 4.22.

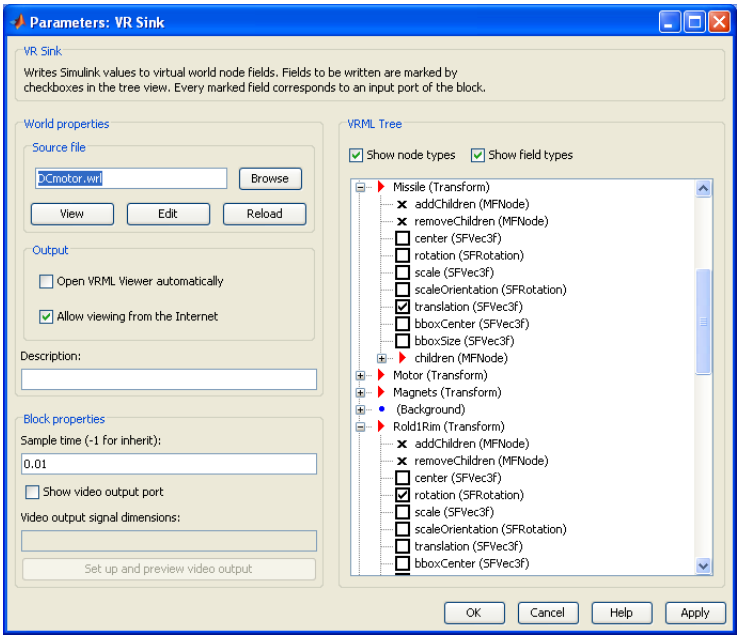

Figura 4.21: Configuração do bloco VR Sink para animação do mundo DCmotor.wrl.

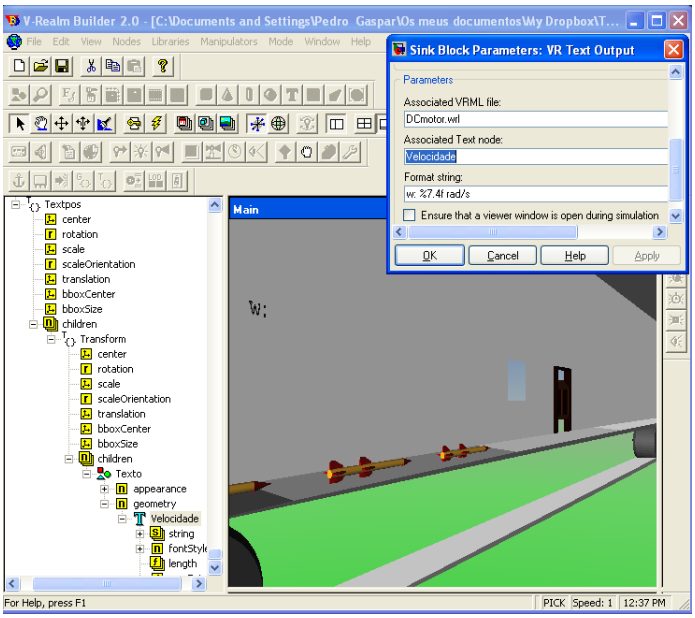

Figura 4.22: Configuração da caixa de diálogo do bloco VR Text Output para controlo do nó de texto.

4.2.5 Projecto de um Controlador PI

É fundamental para o projecto de um controlador fazer reagir o mecanismo por forma a ser atingida a velocidade angular de referência dentro das especificações já referidas. Outro facto importante é compreender qual o sinal de referência com o qual devemos trabalhar. Para este caso assumiu-se uma velocidade linear dos objectos na linha de montagem de 0.15 m/s, o que corresponde a uma velocidade angular de referência de 0.3 rad/s. Posteriormente o sinal de referência é alterado para o dobro, isto é, para $t = 20$ *segundos* assumiu-se um degrau de 0.6 rad/s como velocidade de referência.

Como ponto de partida assumiu-se a topologia de realimentação negativa apresentada no esquema da figura 4.19, cujo o sistema é compensado por um controlador P. Considerando novamente nulas as

condições iniciais, obtém-se a FT do sistema em malha fechada (equação 4.20).

$$CL(s) = \frac{\frac{K_p \cdot K_m}{(Jm+J1)La}}{s^2 + s[\frac{(Jm+J1)Ra+D \cdot La}{(Jm+J1)La}] + \frac{D \cdot Ra + K_m \cdot K_g + K_p \cdot K_m}{(Jm+J1)La}} \tag{4.20}$$

Utilizando as especificações enunciadas para o projecto, calcularam-se dois valores para a constante de ganho proporcional K_p. Um valor foi calculado a partir do tempo de estabelecimento pretendido, usando a expressão $ts(\pm 2\%) \approx \frac{4}{w_n \cdot \xi}$ para a obtenção da frequência natural w_n. Outro valor de K_p foi projectado através das especificações para o erro em regime estacionário. Ambos os valores foram obtidos com base no código no anexo 6.2. Dado que o valor obtido a partir do tempo de estabelecimento é negativo, escolheu-se um controlador P com o maior dos ganhos proporcionais, $K_p = 190.19$ e obtiveram-se os resultados apresentados na figura 4.23.

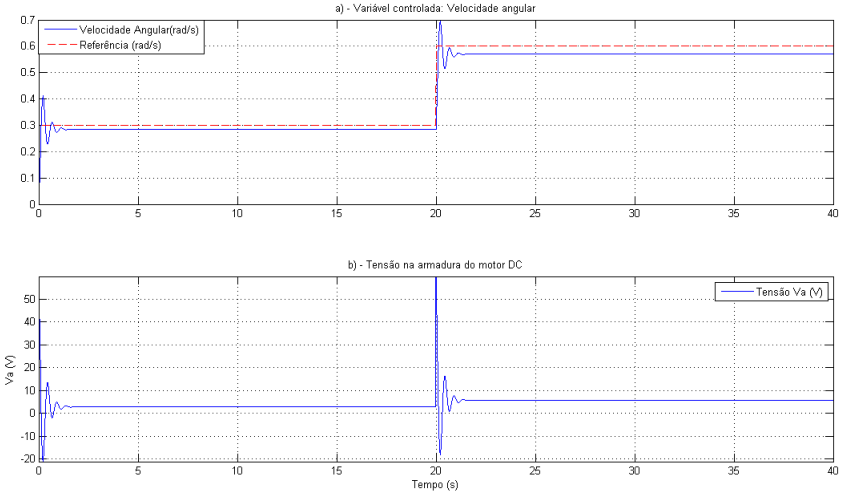

Figura 4.23: Resposta do sistema no tempo com controlador tipo P

Atendendo aos dados obtidos a partir da simulação do modelo linear do conjunto motor DC -linha de montagem, verifica-se que com o valor calculado para a constante de ganho proporcional K_p, as características dinâmicas pretendidas para o sistema não são todas atingidas com sucesso. Neste caso específico observa-se uma sobre-elevação superior aos 10% defnidos, assim como um valor diferente de zero para o erro em regime estacionário. O tempo de estabelecimento é igualmente cumprido mas à custa de variações na tensão Va muito repentinas e de difícil realização física.

De acordo com a tabela 4.1 referida em 4.1.5, e tendo em conta que o modelo linar do sistema não apresenta pólos na origem, consegue-se anular o erro ao introduzir uma componente intregrativa no controlador. Projectando um compensador cuja FT é dada pela equação 4.21:

$$Gc(s) = \frac{C_{PI}(s)}{E(s)} = K_p + \frac{K_i}{s} = \frac{K_p s + K_i}{s} \tag{4.21}$$

onde K_p é a constante de ganho proporcional e K_i a constante de ganho integral, é expectável um comportamento dinâmico que cumpre com as características definidas para a resposta do sistema. Ou seja, diminuindo o valor da constante de ganho proporcional e introduzindo a componente integral no controlador, é possível conjugar valores tal que se diminui a sobre-elevação e se atinge um valor nulo para ess.

Para a determinação das constantes referidas considerou-se a FT em malha fechada $CL(s) = \frac{OL(s) \cdot Gc(s)}{1 + OL(s) \cdot Gc(s)}$. Considerando os coeficientes $a2$, $a1$, $a0$ e b definidos no algoritmo 4.3 obteve-se a seguinte equação:

$$CL(s) = \frac{(K_p \cdot s + K_i) \cdot b}{a2 \cdot s^3 + s^2 \cdot a1 + s \cdot (a0 + b \cdot K_p) + b \cdot K_i} \tag{4.22}$$

A partir da equação 4.22 e partindo da forma genérica de um sistema de 2ª ordem, podemos obter as seguintes equações características:

$$\begin{cases} a2 \cdot s^3 + s^2 \cdot a1 + s \cdot (a0 + b \cdot K_p) + b \cdot K_i = 0 \\ (s^2 + 2\xi\omega_n s + \omega_n^2)(s + \beta) = 0 \end{cases} \tag{4.23}$$

Nas equações presentes no sistema 4.23, de notar que foi introduzido um pólo β na equação característica de um sistema genérico de 2ª ordem. Este deve ser projectado afastado da origem e com pouca influência na resposta do sistema. Assim, podem deduzir-se as seguintes expressões:

$$2\xi\omega_n + \beta = a1 \tag{4.24}$$

$$2\xi\omega_n\beta = b \cdot K_p \tag{4.25}$$

$$K_i \cdot b = a0 \cdot \beta \tag{4.26}$$

Os parâmetros ξ(coeficiente de amortecimento) e ω_n(frequência natural) são obtidos directamente a partir das especificações dinâmicas do sistema, recorrendo às expressões retiradas de [10]. Das igualdades exprimidas pelas equações 4.24, 4.25 e 4.26 retiraram-se os valores base $\beta = 6.2$, $K_p = 4.96$ e $K_i = 62.062$. Usando as técnicas de projecto de controlo sugeridas em [24], projectaram-se as constantes de ganho proporcional $K_p = 24.3$ e constante de ganho integral $K_i = 27.93$, através de um processo iterativo. Com estes valores obtiveram-se os seguintes resultados (figura 4.24) recorrendo ao código no anexo 6.2.

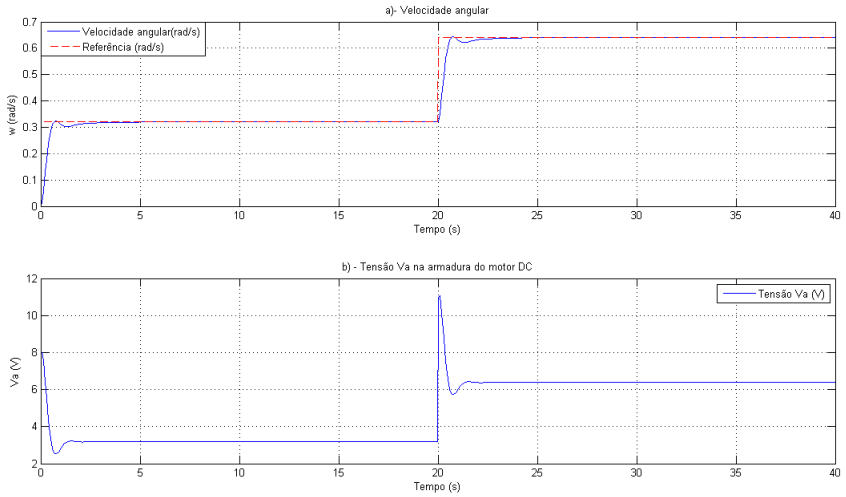

Figura 4.24: Resposta no tempo com compensador tipo PI.

Analisando a figura 4.24 e observando os resultados obtidos executando a ferramenta Sisotool (ver algoritmo 4.4), verifica-se que as especificações do regime transitório e estacionário impostas ao sistema se cumprem com sucesso e com recurso a valores dos parâmetros do controlador, muito inferiores aos obtidos em primeira instância.

No presente caso conclui-se ainda do gráfico b) que a componente integradora do controlador, bem como a redução do valor da constante proporcional produzem uma diminuição na amplitude de Va. Deste modo, Va tende para amplitudes facilmente reproduzidas fisicamente.

Algorithm 4.4 Visualização da resposta do sistema na ferramenta Sisotool com o controlador projectado.

```
%Dados da simulação com PI
%Parâmetros do regime transitório e estacionário no Sisotool
num=[b];
den=[a2 a1 a0];
G=tf(num,den);
s=tf('s');
Gc=(Ki/s+Kp);
Gcl=sisotool(G,Gc)
```

4.2.6 Controlo da velocidade angular do motor DC com compensador cancelamento & implantação de pólo

De modo a contornar a evolução repentina da tensão Va nos terminais da armadura do motor DC, tornando os valores gerados pelo controlador passíveis de obter num sistema físico, procedeu-se ao projecto de um compensador do tipo cancelamento/implantação de pólo. A FT característica deste tipo de compensadores é dada por:

$$\frac{C(s)}{E(s)} = K_c \cdot \frac{s+a}{s+b} \begin{cases} a > 0 \\ b > 0 \end{cases} \tag{4.27}$$

Os parâmetros envolvidos na dinâmica do compensador foram definidos de acordo com os seguinte: a é o módulo do valor mais dominante entre os pólos e zeros do sistema não compensado e $b = 0.1 \cdot a$. O parâmetro K_c é o ganho proporcional total do compensador, o qual permite ajustar a resposta final do sistema.

A execução do *script* Matlab® no anexo 6.3 permite projectar uma resposta do sistema com o compensador representado na equação 4.27. A evolução gráfica dos sinais velocidade angular (ω) e da tensão Va apresentam-se em seguida.

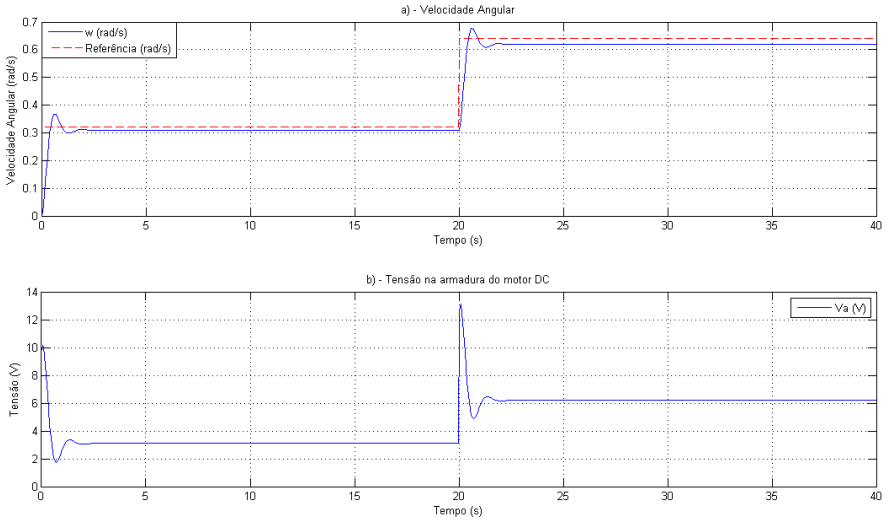

Figura 4.25: Resposta temporal do compensador do tipo cancelamento/implantação de pólo.

Com a introdução de um compensador cancelamento/implantação de pólo e assumindo a constante de ganho $K_c = 1$ para realizar o controlo de um motor DC, verifica-se que o cumprimento das características dinâmicas não é conseguido. A realização física do caso de controlo em questão depende grandemente da tensão de controlo Va. Por este motivo, ajustou-se o valor de K_c do compensador atraso de modo a obter-se uma conjugação de parâmetros que resulte em valores fisicamente realizáveis

para a tensão Va. Com o ajuste da constante de ganho K_c para o valor 30, observa-se que na figura 4.25 o erro em regime estacionário é cumprido, assim como o tempo de estabelecimento definido. À custa da garantia destas especificações não se cumpre o factor de sobre-elevação (PO), mas conseguem-se amplitudes realizáveis para o sinal Va.

4.2.7 Controlo da velocidade angular do motor DC com compensador avanço

Um compensador avanço traduz um comportamento dinâmico contrário ao descrito para um compensador atraso. Sendo o comportamento do compensador avanço idêntico ao de um controlador tipo PD e considerando a análise e estudo deste tipo de compensadores descrito em [24], considerou-se a FT na equação 4.28:

$$G_{av}(s) = \frac{C_{av}(s)}{E(s)} = K_{av} \cdot K_d \cdot \frac{s + \frac{K_p}{K_d}}{s + \lambda \cdot \frac{K_p}{K_d}} \tag{4.28}$$

Como foi já referido, o comportamento de um compensador avanço aproxima-se da resposta de um controlador PD. Deste modo, assumiu-se para efeitos de cálculo base que $K_{av} = 1$ e:

$$G_{av}(s) \approx K_{av} \cdot (K_d \cdot s + K_p) \tag{4.29}$$

$$CL(s) = \frac{(K_p + K_d s) \cdot b}{a2 \cdot s^2 + s \cdot (a1 + b \cdot K_d) + a0 + b \cdot K_p} \tag{4.30}$$

Considerando a fórmula genérica de um sistema de 2ª ordem e a partir da substituição da equação 4.29 na equação 4.18, calcularam-se os valores das constantes de ganho derivativo e proporcional. Após o ajuste das mesmas e o cálculo do valor da constante K_{av} via ferramenta Sisotool (ver código no anexo 6.4), chegou-se à resposta do sistema na figura 4.26:

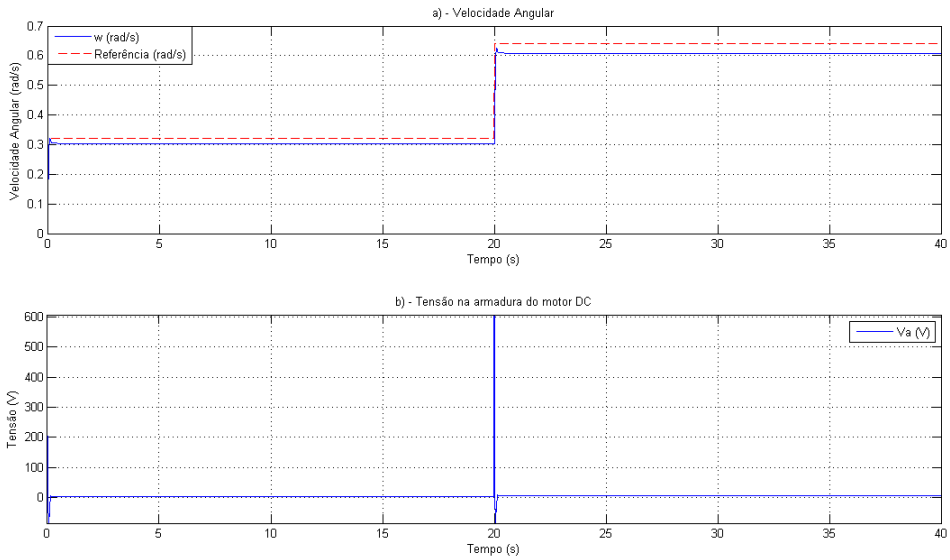

Figura 4.26: Resposta do sistema com compensador avanço.

Como é visível, a resposta do sistema utilizando um compensador avanço consegue suportar as características definidas previamente para a resposta transitória do motor DC. No regime transitório

verifica-se um tempo de subida rápido, bem como um tempo de estabelecimento (ts) abaixo do definido. Graficamente observa-se ainda uma sobre-elevação (PO) dentro dos limites especificados. Em relação ao regime estacionário conclui-se que o erro (ess) definido não é respeitado. Estes resultados obtiveram-se à custa de valores impraticáveis da tensão de controlo Va, dado que a curva em b) denota variações bruscas aquando da alteração da referência. Em suma, a nível físico o controlo deste electro-mecanismo não se realizaria usando um compensador avanço.

No esquema gráfico seguinte apresentam-se as respostas do sistema utilizando os vários tipos de controladores projectados de forma a perceber-se qual o que proporcionou um melhor comportamento dentro das especificações definidas e tendo em conta alguns limites físicos dos componentes do sistema, como por exemplo a tensão Va .

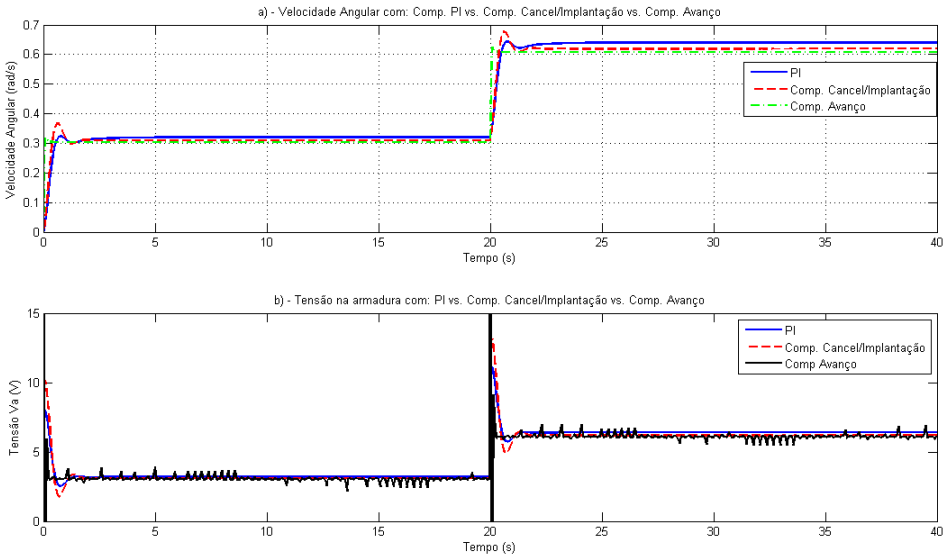

Figura 4.27: Comparação da resposta dada pelos três controladores implementados.

Como conclusão final, pode observar-se pelas curvas nos gráficos que existe alguma proximidade nas respostas do sistema com os vários compensadores. Apesar deste facto, é necessário fazer uma abordagem conjunta às curvas da figura 4.27.

Relativamente às especificações de dinâmica previamente definidas verifica-se que apenas o compensador PI as consegue cumprir na totalidade. O compensador PI é também o único a apresentar um erro em regime estacionário nulo, mas revela-se ligeiramente mais lento nos tempos de subida (tr) e estabelecimento da resposta (ts) comparativamente aos outros compensadores projectados. Em termos de resposta ao sinal de entrada implementado é visível que o compensador cancelamento/implantação de pólo apresenta sobre-elevação, sempre que existe uma alteração repentina relativamente ao estado anterior. Esta sobre-elevação é, ao contrário da medida para o controlador PI, superior à desejada. As outras características da curva respeitantes ao compensador cancelamento/implantação de pólo estão em conformidade com as especificações mencionadas para o regime transitório e estacionário.

Por último, o compensador avanço é o que denota um funcionamento menos adequado. Neste caso verifica-se a resposta mais rápida de entre os três controladores, cumprindo o tempo de estabelecimento definido e ainda o erro em regime estacionário. Por outro lado, observa-se que tem um comportamento demasiado oscilatório aquando da alteração da amplitude do sinal de referência.

No que à tensão Va diz respeito, utilizou-se uma escala logarítmica para se demonstrar a descrepância de valores obtidos para controlar a tensão aos terminais do motor DC. Da análise da figura 4.27 - b) conclui-se que os valores gerados pelo compensador avanço ultrapassam os limites físicos de realização. Assim, e tendo em conta as ilações tiradas anteriormente, verifica-se que o compensador PI é aquele que apresenta um melhor desempenho de acordo com as limitações físicas do motor DC e em concordância com as características dinâmicas definidas para o sistema.

A execução da experiência com a VRT produziu os resultados na figura 4.28.

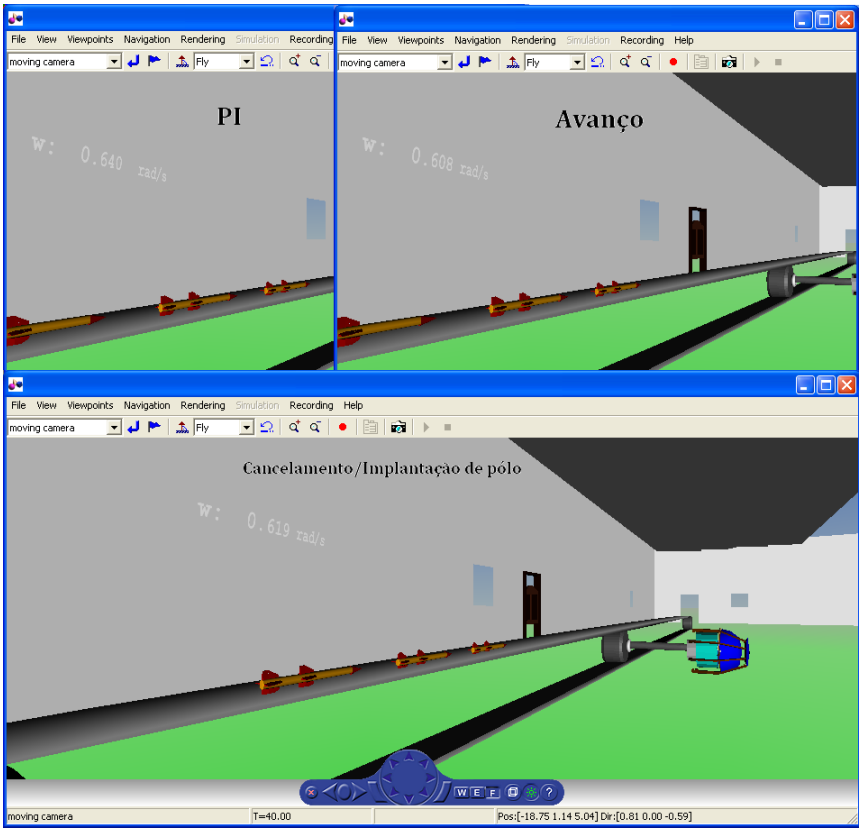

Figura 4.28: Visualização da resposta no modelo 3D para os 3 compensadores.

4.3 Controlo do Nível de Água num Esquema de Dois Tanques Acoplados

4.3.1 Introdução

O terceiro caso de estudo pretende controlar o nível de água num tanque de um sistema de dois de tanques acoplados por um tubo que oferece resistência ($R1$) à passagem do fluído. Neste exercício é considerado um caudal (q) como entrada do sistema, sendo esta a variável sobre a qual se vai actuar de modo a atingir uma altura de água ($H2$) pré-definida. Para a realização e visualização desta experiência implementou-se um ambiente virtual de 2 tanques ($T1$ e $T2$) de dimensões fixas, recorrendo, tal como para os casos anteriores, à ferramenta V-Realm Builder 2.0. O controlo deste modelo virtual foi realizado através de um modelo matemático linear projectado em Simulink®. O esquema do mecanismo que se pretende controlar está patente na figura 4.29.

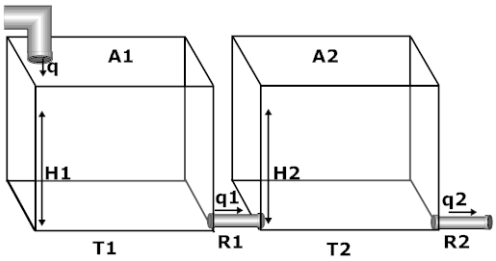

Figura 4.29: Esquema de funcionamento do sistema de dois tanques acoplados.

De acordo com o esquema do sistema a desenvolver há ainda a necessidade de referir que o caudal

(q) fornecido ao tanque ($T1$) é controlado por uma bomba. Após pesquisa efectuada, verificou-se que os valores máximos de caudal rondam os $0.1\,m^3/s$. Assim, para este caso considerou-se a saturação do caudal (q) para um valor de $0.2\,m^3/s$. Por outro lado, verifica-se que o tanque ($T2$) apresenta um tubo de escape de água com resistividade $R2$ que vai influenciar também o cumprimento da altura $H2$ pré-definida.

As variáveis que modelam o sistema descrevem-se em seguida:

- $q1$ - Caudal de água que flui do tanque $T1$ para o tanque $T2$
- $q2$ - Caudal de água que flui do Tanque $T2$ para o exterior
- $R1$ e $R2$ - Resistência oferecida à passagem, respectivamente, dos caudais $q1$ e $q2$
- $H1$e $H2$ - Altura de água nos tanques $T1$ e $T2$, respectivamente
- $A1$e $A2$ - Área da base dos tanques
- q - Caudal de entrada no tanque $T1$

A compensação do sistema com o fim de se atingir a altura $H2$ pré-definida tem que ser realizada respeitando algumas características dinâmicas. Neste caso, os controladores projectados basearam-se nas seguintes especificações:

- Regime transitório:

1. Sobre-elevação (PO) máxima de 8.5% relativo ao valor final;
2. Tempo de estabelecimento (ts) de 50 segundos;

- Regime estacionário:

1. Erro em regime estacionário (ess) de 0.1 relativamente ao valor de referência;

4.3.2 Cálculo do Modelo Matemático do Sistema

Adicionalmente às variáveis descritas anteriormente, para se projectar o modelo matemático linear que representa a dinâmica do esquema apresentado na figura 4.29 é necessário considerar os efeitos da gravidade ($g = 9.8\,m/s^2$) e da densidade do fluído com o qual estamos a lidar ($\rho = 1000\,Kg/m^3$).

Tendo em conta o sistema hidráulico no esquemático da figura 4.29 partiu-se da definição de pressão (equação 4.31) para chegar ao modelo matemático que representa o sistema.

$$P = \frac{\rho V g}{A} = \frac{Ah\rho g}{A} = \rho g h \tag{4.31}$$

Assim,

$$q1(t) = \frac{\rho g \cdot h1(t) - \rho g \cdot h2(t)}{R1} \tag{4.32}$$

e

$$q2(t) = \frac{\rho g \cdot h2(t)}{R2} \tag{4.33}$$

O caudal final em cada um dos tanques ($qf1$ e $qf2$) é definido como a variação entre o caudal de entrada e o caudal de saída.

$$qf1 = A1 \cdot \frac{dh1(t)}{dt} = q(t) - q1(t) \tag{4.34}$$

$$qf2 = A2 \cdot \frac{dh2(t)}{dt} = q1(t) - q2(t) \tag{4.35}$$

Após substituição das equações 4.32 e 4.33 nas expressões obtidas para os caudais $qf1$ e $qf2$, chegou-se ao seguinte sistema de equações no domínio de Laplace:

$$\begin{cases} H1(s)\cdot s = & H1(s)\cdot \frac{-\rho g}{R1\cdot A1} + H2(s)\cdot \frac{\rho g}{R1\cdot A1} + Q(s)\cdot \frac{1}{A1} \\ H2(s)\cdot s = & H1(s)\cdot \frac{\rho g}{A2\cdot R1} + H2(s)\cdot \frac{-\rho g}{A2}\left(\frac{R1+R2}{R1\cdot R2}\right) \end{cases} \tag{4.36}$$

Em seguida definiram-se os coeficientes $a0 = \frac{-\rho g}{R1\cdot A1}$, $a1 = \frac{\rho g}{A2\cdot R1}$, $a2 = \frac{-\rho g}{A2}\left(\frac{R1+R2}{R1\cdot R2}\right)$ e $b = \frac{1}{A1}$, por forma a reduzir a complexidade dos cálculos.

Considerando uma abordagem clássica ao sistema de controlo em análise, desenvolveu-se o modelo matemático linear que relaciona a variável de controlo $H2(s)$ e a entrada do sistema $Q(s)$. Considerando nulas as condições iniciais do sistema, ou seja, assumindo que ambos os tanques não apresentam qualquer nível de fluído inicialmente, chegou-se à FT na equação 4.37.

$$G(s) = \frac{H2(s)}{Q(s)} = \frac{b\cdot a1}{s^2 + s\cdot(a0+a2) + a0\cdot a2 - a0\cdot a1} \tag{4.37}$$

Analogamente, e a partir do sistema de equações 4.36, obteve-se também a relação entre a altura de fluído $H1(s)$ no tanque $T1$ e o caudal de entrada $Q(s)$.

$$G2(s) = \frac{H1(s)}{Q(s)} = \frac{b\cdot s + a2\cdot b}{s^2 + s\cdot(a0+a2) + a0\cdot a2 - a0\cdot a1} \tag{4.38}$$

A definição das constantes e dos coeficientes já mencionados foi feita a partir de um *script* Matlab® e os seus valores foram retirados e adaptados de [2]. Um excerto do *script* criado é apresentado no algoritmo 4.5.

Algorithm 4.5 Inicialização das constantes e dos coeficientes do sistema.

```
%Constantes do sistema
ro=1000; %Densidade (kg/m^3)
g=9.8; %Aceleração da gravidade (m/s^2)
A1=1; %Área de secção do tanque 1 real (m^2)
A2=1; %Área de secção do tanque 2 real (m^2)
R1=10e+1; %Resistividade oferecida pelo tubo entre T1 e T2
R2=1000e+2;%Resistividade oferecida pelo tubo entre T2 e o exterior
%Constantes Ambiente virtual
mv=vrworld('watertanks.wrl');
x=nodes(mv,'-full');
hr=mv.Tank1.scale(1,2);%Altura virtual dos tanques
%Coeficientes para cálculo da função de transferência
b=1/A1;
a0=ro*g/(A1*R1);
a1=ro*g/(A2*R1);
a2=ro*g*(R1+R2)/(A2*R1*R2);
```

4.3.3 Construção do Ambiente 3D

A operação do sistema de dois tanques acoplados vai ser visualizada no ambiente 3D implementado usando a ferramenta V-Realm Builder 2.0. Para o sistema de controlo descrito anteriormente usaram-se os objectos das bibliotecas fornecidas pela aplicação para a construção de um mundo virtual onde fosse possível visualizar a progressão no tempo do nível de fluído em ambos os tanques.

Tal como nos casos de estudo anteriores, implementou-se neste mundo virtual um painel de texto que permite acompanhar a evolução da variável de controlo ao longo do tempo.

O resultado obtido é apresentado na figura 4.30.

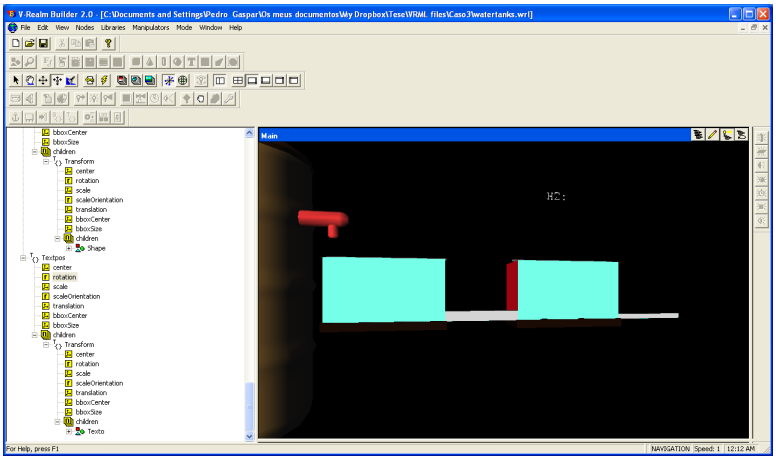

Figura 4.30: Ambiente virtual implementado para simulação do sistema de 2 tanques acoplados.

4.3.4 Integração do Ambiente 3D com a Simulink® - VRT

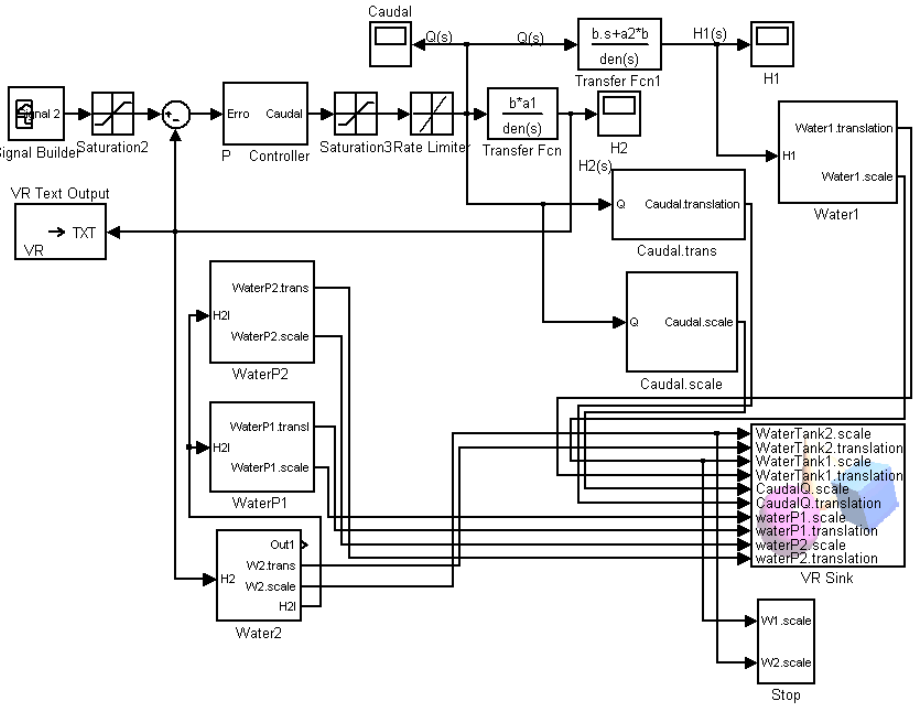

Figura 4.31: Diagrama de blocos em Simulink® para integração do modelo virtual com a VRT.

Após o projecto do modelo 3D e a modelação matemática do sistema, há a necessidade de os fazer interagir. A interacção entre estes dois ambientes foi realizada a partir do diagrama de blocos construído no Simulink® e visível na figura 4.31.

É importante frisar que inicialmente foi feita uma abordagem em termos de controlo clássico, ou seja, o primeiro passo passou por definir a estratégia de controlo a aplicar ao sistema. Como se pode observar na figura abaixo optou-se por se realizar a compensação do sistema, utilizando um controlador do tipo P. Para esta estratégia utilizaram-se os blocos "Transfer Fcn" e "Transfer Fcn1" para se obter respectivamente, a altura $H1$ e $H2$ em função do caudal q gerado pelo controlador. Outro factor considerado foi o caudal máximo possível, o qual foi definido como 0.2 m^3/s, e ainda a variação máxima permitida no fornecimento deste caudal. De acordo com os limites físicos descritos

anteriormente utilizaram-se os blocos "Saturation3" e "Rate Limiter".

A ligação do modelo matemático linear com os blocos da VRT e consequentemente com o mundo virtual arquitectado, foi feito através dos subsistemas presentes no diagrama de blocos da figura 4.31. Alguns destes subsistemas funcionam de forma análoga. O facto de funcionarem de maneira semelhante justifica-se por controlarem as mesmas características dos objectos da cena 3D, mas de nós diferentes.

4.3.4.1 Controlo da escala e deslocamento da água nos tanques

Para se visualizar o efeito de enchimento dos tanques utilizaram-se duas massas de água (uma para cada um dos tanques). A altura destas massas de água é definida directamente a partir dos blocos "Transfer Fcn" e "Transfer Fcn1" que geram ao longo do tempo as alturas $H2$ e $H1$. A transdução dos valores gerados para valores compatíveis com o cenário projectado em 3D é feito através do bloco "Water2", cujo funcionamento é análogo ao bloco "Water1". Na figura 4.32 é apresentada a máscara do primeiro, que apresenta como única diferença relativamente ao segundo o sinal de entrada.

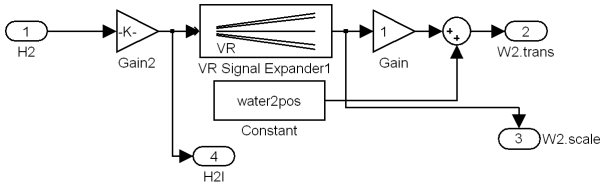

Figura 4.32: Diagrama de blocos de controlo da escala e translação da massa de água no tanque $T2$.

O bloco de ganho "Gain2" é responsável pela relativização da altura de água gerada de acordo com a altura máxima dos tanques no ambiente virtual. Como foi referido, este cálculo é realizado através da divisão do valor real gerado nos blocos do tipo "Transfer Fcn", pelo valor da constante hr. O sinal de saída do bloco "Gain2" é seguidamente passado ao já conhecido "VR Signal Expande1", o qual preenche o campo "WaterTank2.translation" com a altura calculada através da FT. Para que apenas se interfira na altura da massa de água o bloco "VR Signal Expande1" deve ser configurado como a figura 4.33 demonstra.

O sinal de saída do bloco "VR Signal Expande1" foi utilizado para controlar directamente a escala no eixo Oy da massa de água, mas também para gerar o sinal necessário ao deslocamento. Este último é obtido pela soma do sinal na saída do bloco "VR Signal Expande1" com a posição inicial da massa (blobo "Constant") de água na cena 3D.

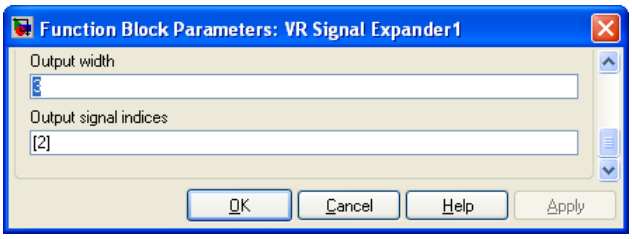

Figura 4.33: Configuração do bloco VR Signal Expander1.

4.3.4.2 Controlo do nível de fluído nos tubos de ligação

Tal como nos subsistemas anteriores, também os blocos "WaterP1" e "WaterP2" funcionam de forma semelhante. O primeiro é responsável por controlar o nível de água no tubo de passagem entre o tanque $T1$ e o tanque $T2$. O segundo controla o nível fluído que flui do tanque $T2$ para o exterior. Como os tubos referidos apresentam diferentes capacidades de passagem de água, existe a necessidade de controlar o nível visualizado de forma diferente. Na figura 4.34 é apresentado o esquema desenvolvido em Simulink para realizar o controlo da escala e deslocamento do fluído nos tubos de passagem de $q1$ e $q2$.

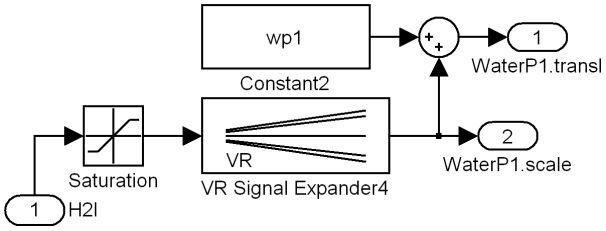

Figura 4.34: Diagrama de blocos de controlo da escala e translação do caudal $q1$ do tanque $T1$ para o tanque $T2$.

A diferença no funcionamento dos blocos "WaterP1" e "WaterP2" encontra-se na definição dos limites de saturação do bloco "Saturation". Para o tubo de resistência $R1$ definiu-se o limite superior de saturação como $R1sat - 0.02$, enquanto que para o tubo de resistência $R2$ se definiu o limite superior de saturação $R1sat/2 - 0.01$. A constante $R1sat$ é obtida a partir da leitura do campo "R1.scale(1,2)" (ver anexo 6.5).

O bloco "VR Signal Expander4" foi configurado da mesma forma que no ponto 4.3.4.1.

O funcionamento dos restantes blocos deste subsistema assemelha-se ao descrito no ponto 4.3.4.1, diferenciando-se apenas pelo valor que é definido no bloco "Constant2".

4.3.4.3 Controlo do caudal fornecido ao tanque $T1$

Para o controlo do caudal a ser fornecido pelo controlador implementado foram utilizados dois subsistemas: um para controlar a escala e outro para controlar o deslocamento.

O bloco "Caudal.scale" contém o subsistema responsável por simular a variação na escala do caudal. Este subsistema é apresentado na figura 4.35.

O caudal é fornecido através de uma torneira. Como tal, a largura e a profundidade do caudal são, no máximo, iguais às dimensões da boca de saída da torneira. No subsistema apresentado, as dimensões máximas são cumpridas devido ao uso do bloco "Saturation4". Para a simulação da água fornecida ao tanque $T1$ foi assumida constante a profundidade do caudal, ou seja, a componente de escala no eixo Oz tem sempre o seu valor máximo, sendo este definido pelo bloco "Constant4". Assim, restam apenas as componentes de altura e largura para controlar de acordo com o caudal q gerado pelo compensador.

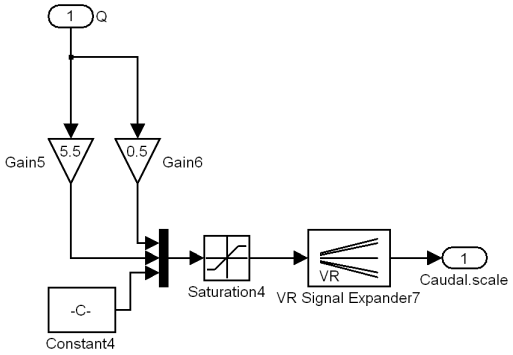

Figura 4.35: Controlo da escala do caudal q .

A partir do sinal q arquitectou-se uma correlação entre a escala no eixo Ox e o eixo Oy tendo em conta não só as dimensões máximas do objecto na cena 3D, mas também o valor máximo possível para o caudal. A correlação projectada é realizada através dos blocos de ganho "Gain5" e "Gain6". O primeiro adequa a altura do caudal e o segundo faz a transdução do valor de caudal para largura.

O sinal a ser passado ao bloco "VR Signal Expander7" foi criado através de um vector com os valores gerados pelos blocos de ganho e pelo bloco "Constant4". Este vector criou-se a partir do bloco

"Mux". Por fim, a configuração do bloco "VR Signal Expander7" traduz-se no resultado apresentado na figura 4.36.

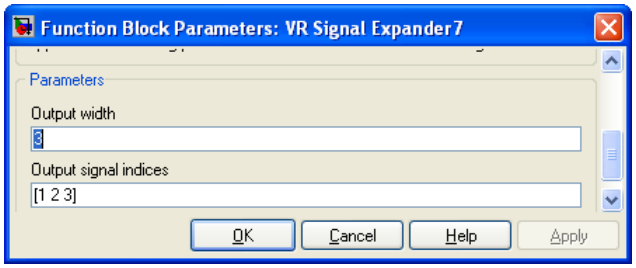

Figura 4.36: Caixa de diálogo de configuração do bloco VR Signal Expander7 .

Para se controlar o deslocamento do caudal usou-se o bloco "Caudal.trans" da figura 4.31. Se se observar debaixo da máscara deste subsistema podemos encontrar os blocos presentes na figura 4.37.

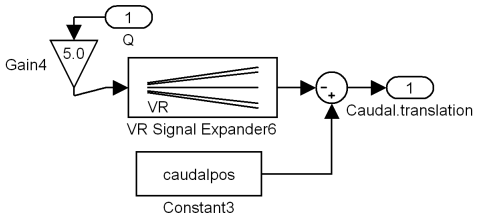

Figura 4.37: Controlo da tanslação do caudal q.

O valor de q gerado ao longo do tempo não permite ser usado directamente para se simular o débito de água de uma torneira. Por este motivo é usado um bloco de ganho "Gain4". Com este bloco foi possível transformar o valor de caudal q para uma translação no sentido do tanque $T1$.

Ao contrário das massas de água que simulam o enchimento dos tanques e se deslocam no sentido positivo do eixo Oy, o caudal move-se no sentido contrário. Ou seja, neste caso a altura gerada não é acrescentada à posição inicial mas subtraída a esta depois de passar pelo bloco "VR Signal Expander6". Este é configurado de forma semelhante ao descrito em 4.3.4.1.

4.3.4.4 Controlo do bloco de texto

O bloco "VR Text Output" é controlado directamente a partir do sinal $H2$. Este bloco permite acompanhar a evolução da altura de fluído no tanque $T2$ ao longo do tempo. Para que este apresentasse o funcionamento correcto acrescentou-se um nó de texto à cena construída no V-Realm Builder 2.0 e configurou-se o bloco da VRT como exemplifica a figura 4.38.

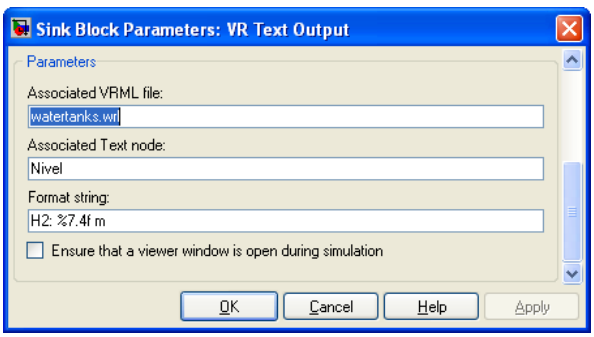

Figura 4.38: Configuração do bloco VR Text Output para visualização da altura de água no tanque $T2$.

4.3.5 Projecto de controlador P

Neste ponto pretende projectar-se um compensador que faça o sistema reagir ao sinal de referência e à sua alteração de acordo com as especificações descritas em 4.3.1. Para além disso, consideraram-se duas perturbações ao sinal de saída, sendo estas apresentadas adiante conjuntamente com o comportamento gráfico do sistema.

Considerando o esquema apresentado na figura 4.31, projectou-se o compensador do tipo P como se apresenta em seguida:

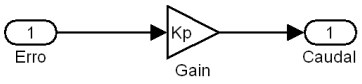

Figura 4.39: Controlador tipo P (Proporcional).

O valor da constante de ganho proporcional K_p foi calculada com base na ferramenta Sisotool do Matlab®. Para isso, executou-se o código no anexo 6.5 e em seguida ajustou-se o valor da constante de ganho proporcional recorrendo a várias tentativas. Com o valor $K_p = 25$ testou-se o sistema e obtiveram-se os seguintes resultados:

Como se pode observar na figura 4.40 no gráfico *a*), quer o tempo de estabelecimento, quer a sobre-elevação definida são cumpridos. Pode assim concluir-se que é possível controlar o nível de água no tanque $T2$ apenas com um controlador proporcional. Outra conclusão relevante a que se chegou, foi que com este compensador não se ultrapassaram os limites físicos de débito de fluído, isto é, consegue-se atingir o nível de água pré-definido sem que se ultrapasse o caudal máximo de $0.2\,m^3/s$, como demonstra a curva do gráfico *b*). Em regime estacionário verifica-se também a obtenção de um sinal de erro inferior ao imposto.

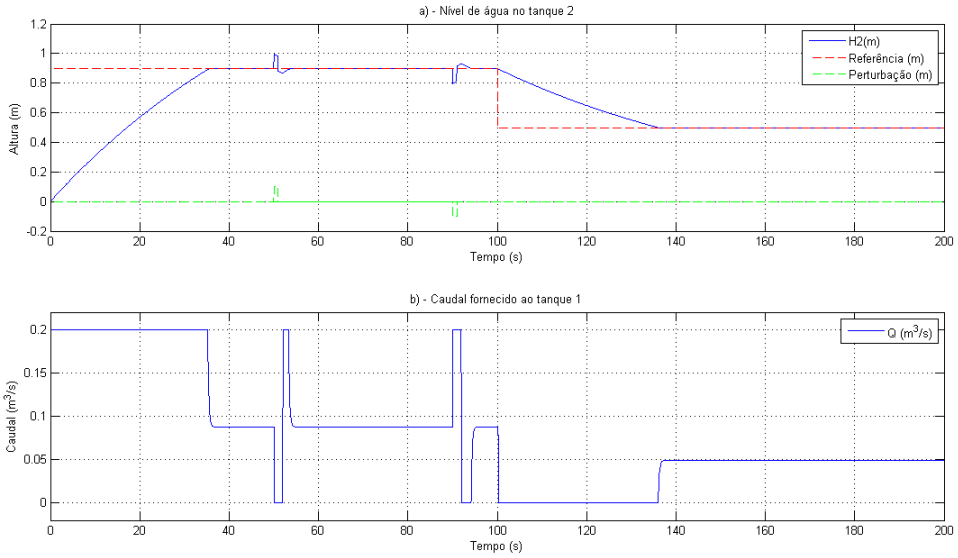

Figura 4.40: Altura de água no tanque $T2$ e caudal q debitado com controlador tipo P.

4.3.6 Arquitetura de um controlador PI

Apesar de se conseguirem cumprir as especificações de dinâmica para o regime transitório e estacionário do sistema em análise a partir da introdução de um compensador tipo P, consegue-se anular o erro em regime estacionário através de um controlador PI. Desta forma realizam-se mais facilmente os valores das constantes de ganho integral e proporcional a nível físico. Estas ilações resultam do facto de este ser um sistema que não apresenta pólos na origem.

Para o projecto do controlador PI partiu-se da FT em malha fechada do sistema $CL(s)$ (equação 4.40) e calcularam-se os valores base para as constantes de ganho proporcional e integrativo.

$$Gc(s) = \frac{C_{PI}(s)}{E(s)} = K_p + \frac{K_i}{s} = \frac{K_p s + K_i}{s} \tag{4.39}$$

$$CL(s) = \frac{(K_p \cdot s + K_i) \cdot (b \cdot a1)}{s^3 + s^2 \cdot (a0 + a2) + s \cdot (a0 \cdot a2 - a0 \cdot a1 + K_p \cdot b \cdot a1) + K_i \cdot b \cdot a1} \tag{4.40}$$

Tal como havia sido feito para a compensação do motor DC com um controlador PI (ponto 4.2.5), usou-se a fórmula genérica de sistemas de 2ª ordem sem zeros, adaptada para sistemas de 3ª ordem. Ou seja, introduz-se um pólo real β afastado do eixo imaginário e com pouca influência na dinâmica do sistema. De acordo com o referido obtém-se a equação 4.41.

$$G(s) = \frac{A \cdot w_n^2}{s^3 + s^2 \cdot (\beta + 2 \cdot \xi \cdot w_n) + s \cdot (2 \cdot \xi \cdot w_n \cdot \beta + w_n^2) + w_n^2 \cdot \beta} \tag{4.41}$$

Igualando os denominadores de $G(s)$ e $CL(s)$ calculam-se facilmente os valores base para as constantes K_p e K_i. Depois de calculados, estes foram ajustados através de um processo iterativo e tendo como base de acção a tabela referida em 4.1.5 e as técnicas aprendidas em [24]. Executando o *script* no anexo 6.5 obteve-se $K_p \approx 2.44$ e $K_i \approx 0.0061$ obtiveram-se os resultados seguintes:

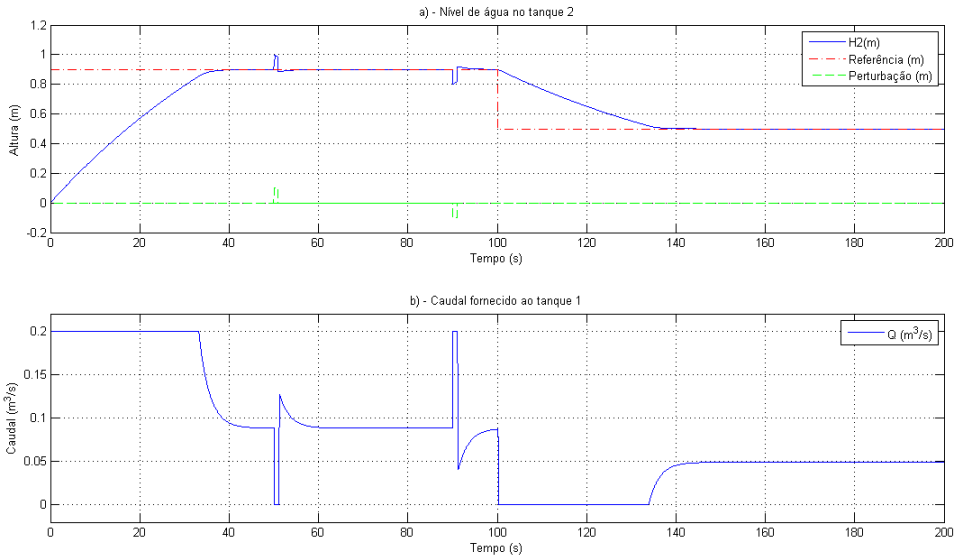

Figura 4.41: Nível de água no tanque $T2$ e caudal q gerado pelo controlador PI

De notar que o valor do pólo real β (14.85) foi também ajustado de forma a que o seu efeito fosse o menor possível, no que ao comportamento dinâmico do sistema diz respeito.

Ao observar as curvas de desempenho, em simultâneo, dos dois controladores projectados (imagem 4.42), verificam-se algumas particularidades que os diferenciam. Em termos de rapidez da resposta observa-se na figura 4.42-a) que o compensador PI é ligeiramente mais lento. O factor de sobre-elevação não é denotado em nenhum dos casos (exceptuando os momentos em que as perturbações são assimiladas pelo sistema), sendo por isso um factor desprezável na comparação entre os dois. Noutra perspectiva, observa-se que o erro é nulo utilizando o controlador PI, enquanto com P isso não se verifica. A resposta mais lenta apresentada pelo segundo controlador projectado justifica-se pela forma como é controlado o caudal q. Da figura 4.42-b) pode concluir-se que PI produz um caudal passível de realização prática, ao contrário de P que requer que se actue duma forma repentina no caudal a fornecer ao tanque $T1$ (ou seja, fechar ou abrir as válvulas/torneiras instantaneamente), acção esta de difícil execução prática.

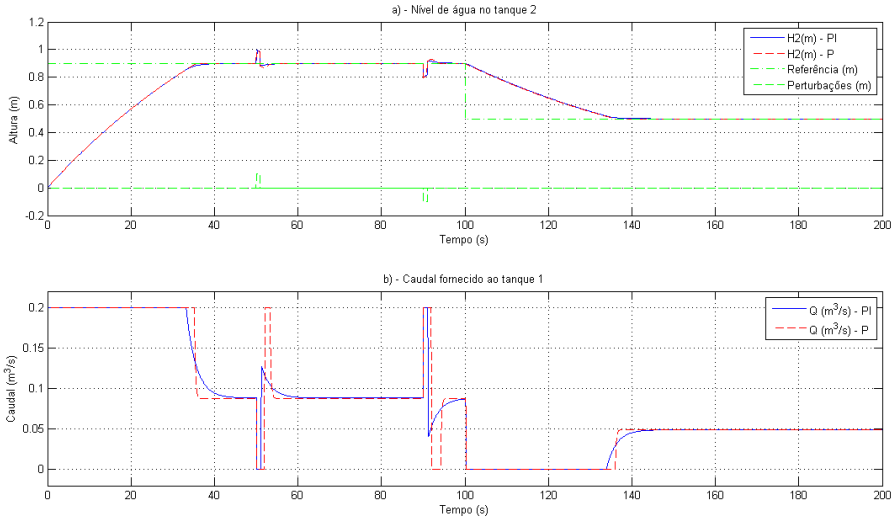

Figura 4.42: Resposta do sistema com controlador P e PI.

Na execução e simulação dos modelos no ambiente virtual é possível verificar a diferença das respostas dos diferentes controladores. Este facto é possível devido ao painel de texto implementado, pois denota a capacidade do controlador PI anular o erro em regime estacionário, enquanto o compensador P mantém apenas uma grande proximidade relativamente ao valor de referência.

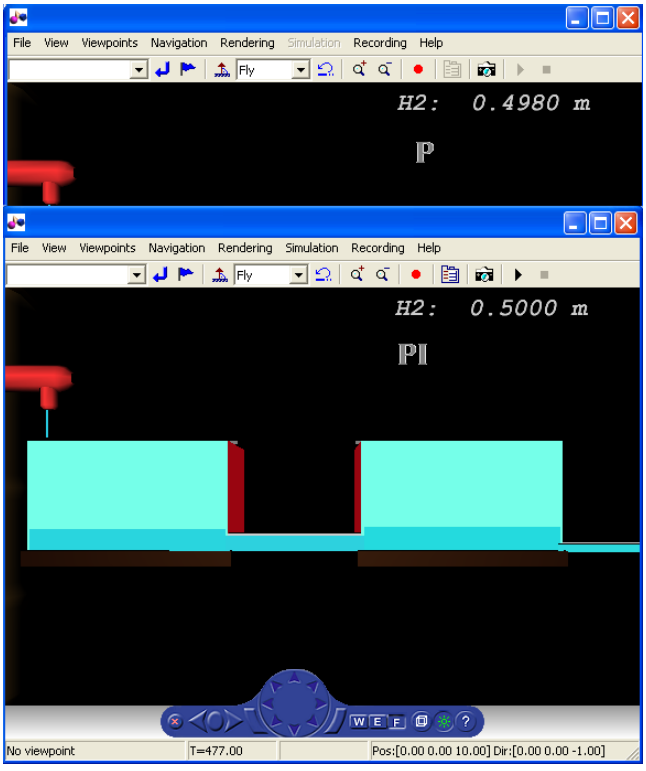

Figura 4.43: Estado do ambiente virtual em regime estacionário.

4.3.7 Projecto de servo controlo

Considerando a complexidade existente nos sistemas de controlo moderno, verifica-se que estes podem ter várias entradas e várias saídas, as quais podem estar inter-relacionadas de uma forma extremamente complicada. Para a análise de um sistema deste tipo é necessário reduzir a complexidade matemática das expressões que regem a dinâmica do sistema. Depois de reduzida a complexidade é possível resolver o controlo do mecanismo à custa de fórmulas computorizadas, evitando assim a tarefa penosa que é o cálculo matricial.[31]

Enquanto o controlo clássico se baseia nas relações entrada-saída através de FT, o controlo moderno parte da descrição do sistema através de n equações diferenciais de 1ª ordem. Estas podem depois ser combinadas, criando uma equação diferencial de 1ª ordem na forma de um vector ou matriz.

Neste ponto é usado a técnica de projecto de compensadores, via implantação de pólos, para um servo sistema do tipo SISO (*Single Input - Single Output*), em detrimento de uma abordagem do tipo regulador a zero, que não garante o cumprimento das especificações em regime estacionário[28].

Da representação obtida no sistema de equações 4.36 e a partir dos coeficientes $a0$, $a1$, $a2$ e b definidos anteriormente, obteve-se a seguinte representação em espaço de estados:

$$\begin{cases} X(s) \cdot s = \begin{bmatrix} -a0 & a0 \\ a1 & -a2 \end{bmatrix} \cdot X(s) + \begin{bmatrix} b \\ 0 \end{bmatrix} \cdot R(s) \\ C(s) = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot X(s) + \begin{bmatrix} 0 \end{bmatrix} \cdot R(s) \end{cases} \tag{4.42}$$

Relativamente aos resultados expressos pela representação em espaço de estados na equação 4.42 considerou-se $X(s) = \begin{bmatrix} H1(s) \\ H2(s) \end{bmatrix}$ e $R(s) = Q(s)$.

O sistema a compensar é do tipo 0, ou seja, ao não possuir qualquer pólo na origem é necessário uma componente integral para anular o erro em regime estacionário. Deste modo usou-se a configuração do diagrama de blocos na figura 4.44.

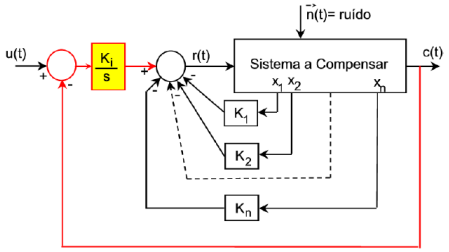

Figura 4.44: Servomecanismo - sistema a compensar do tipo 0 ou superior (retirado de [28])

Para o cálculo dos valores da matriz de realimentação K e do ganho ki recorreu-se ao excerto de código do anexo 6.6 e apresentado no algoritmo 4.6. Este algoritmo demonstra a execução dos passos citados em [3] para a compensação de um servomecanismo cuja entrada de referência é um degrau. Os procedimentos passam por: obter uma representação do sistema em espaço de estados, verificar a controlabilidade relativamente às matrizes A e B, determinar a representação em espaço de estados do sistema original mais o integrador ($Atil$, $Btil$, $Dtil$, 0), verificar a controlabilidade de $Atil$ e $Btil$, projectar o compensador a zero $Ktil = [K \ -ki]$ para ($Atil$, $Btil$, $Dtil$, 0), calcular as matrizes do sistema em malha fechada (Acl, Bcl, Dcl, 0) e verificar a resposta ao degrau do sistema em malha fechada.

Depois de executado o código no anexo 6.6 foi possível verificar que a resposta do sistema ao degrau unitário se adequa às especificações dinâmicas pretendidas para o sistema. A resposta pode ser consultada na figura 4.45.

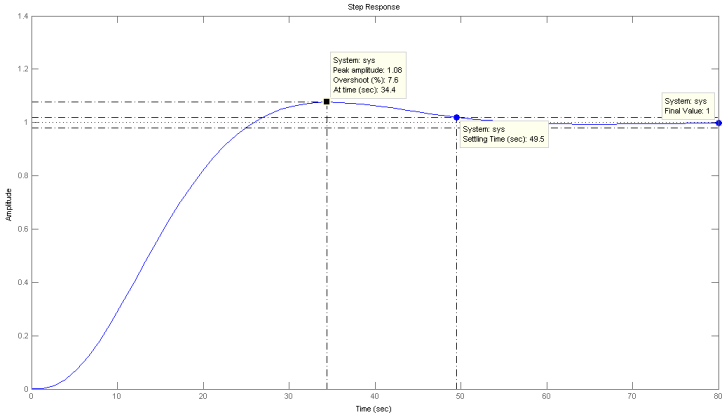

Figura 4.45: Resposta da compensação por servomecanismo na ferramenta Sisotool.

Em Simulink® implementaram-se os dois tanques acoplados a partir do sistema de equações 4.36. Esta realização permitiu ter acesso às duas variáveis de estado individualmente, possibilitando a sua realimentação através da matriz K. De notar que neste caso assumiu-se a possibilidade de existir uma perturbação ao caudal existente no tanque $T2$.

Algorithm 4.6 Algoritmo de controlo usado para o cálculo da matriz de realimentação K e do ganho integrativo ki.

```
%Matrizes da representação em SS
A=[-a0 a0;a1 -a2];
B=[b0;0];
D=[0 1];
E=[0];
%Verificação da controlabilidade
N = [A B; -D 0];
nn = length(A) + 1 - rank(N); % Controlável se nn = 0
% As matrizes dos coeficientes e de entrada:
Z = zeros(length(A),1);
Atil = [A Z; -D 0];
Btil = [B;0];
Dtil = [D 0];
%Controlabilidade do sistema original mais o integrador
Stil=ctrb(Atil,Btil);
nctil = length(Atil)-rank(Stil) % Controlável se nctil = 0
% Especificações para determinação da equação característica do sistema compensado
PO=0.085;
tss=50; %tempo de estabelecimento
ess=0.1;
qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
wn=4/(qsi*tss);
poli=[1 2*qsi*wn wn^2];
polosw=roots(poli);
Pc =[polosw' min(real(polosw))*4]; % Pólos desejados + 1 pólo colocado 4 vezes mais à esquerda
% Matriz de ganho do sistema compensado
Ktil = place(Atil,Btil,Pc);
% Ganho do compensador integrador
ki = -Ktil(:,end);
K = Ktil(:,1:end-1);
```

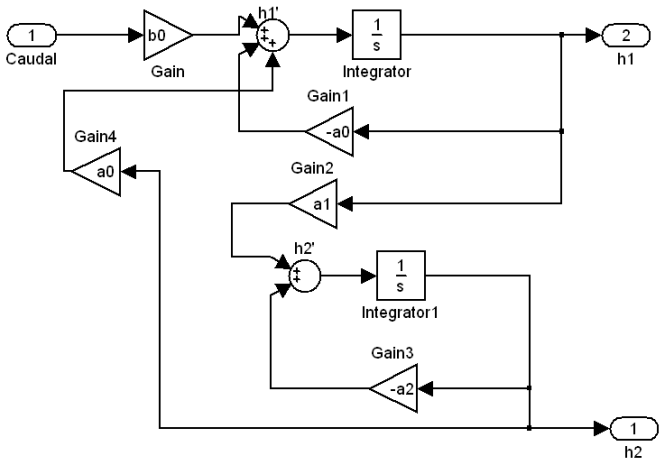

Figura 4.46: Implementação da representação em espaço de estados obtida em Simulink®.

O esquema apresentado na figura 4.46 foi colocado no bloco "SS de 2 tanques acoplados", apresentado na figura 4.47, obtendo-se acina a representação em diagrama de blocos para o sistema implementado:

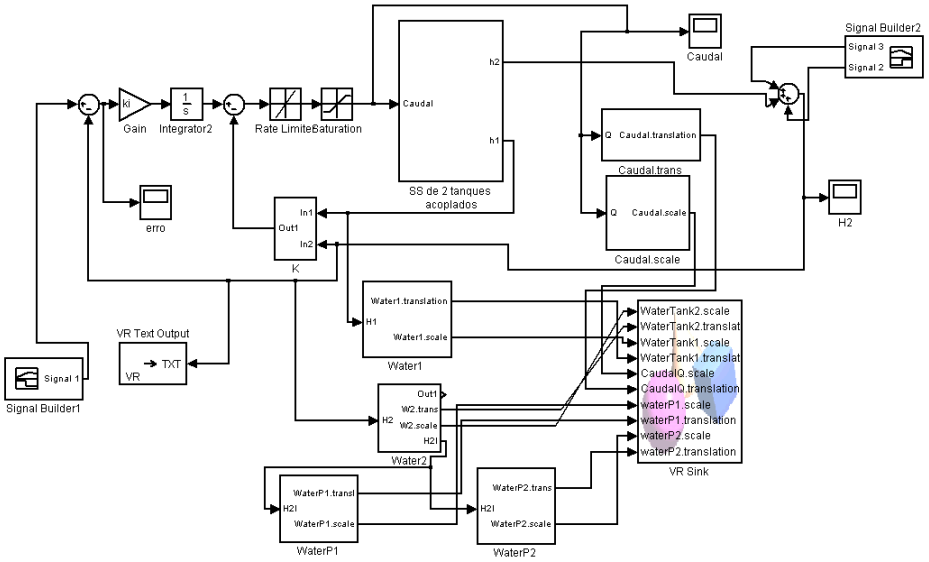

Figura 4.47: Implementação do servo controlo e integração com Simulink®, VRT e o modelo 3D.

Os blocos funcionais são análogos aos descritos em 4.3.4, aquando da implementação do esquema de controlo usando técnicas de controlo clássico. No caso específico do projecto de um servomecanismo usou-se adicionalmente o bloco "K" apresentado na figura 4.48 e a configuração integradora implementada pelos blocos "Gain" e "Integrator2".

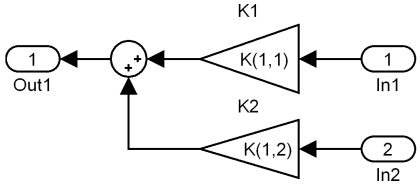

Figura 4.48: Matriz de realimentação K implementada em Simulink®.

A simulação do modelo virtual 3D foi efectuada usando como sinal de referência um bloco "Signal Builder" com alteração da referência de $0.5\,m$ para $1\,m$ em $t = 150$ segundos, e um sinal de perturbação com uma amplitude de $0.1\ m^3/s$ com duração $\tau = 1\,s$ em $t = 60\,s$ e uma amplitude de $-0.1\ m^3/s$ com duração $\tau = 1\,s$ em $t = 90\,s$. Com o projecto de um compensador para servomecanismos obtiveram-se as curvas na figura 4.49.

A análise dos gráficos permite concluir que o projecto de servo sistemas foi realizado com sucesso. Em a) verifica-se que o sistema responde ao degrau de referência de acordo com as especificações em regime transitório, bem como à alteração de referência. Na resposta às perturbações causadas no nível de água no tanque $T2$ observa-se que o sistema atinge os valores de referência novamente. No entanto, neste caso específico o factor de sobre-elevação pré-definido é ultrapassado.

Em regime estacionário observa-se também que o erro (ess) é anulado pela configuração integradora apresentada no esquema 4.44.

Relativamente à evolução da curva representativa do caudal q, pode observar-se na figura 4.49-b) que esta apresenta um comportamento de realização prática possível, não apresentando valores de caudal superiores aos limites físicos praticado.

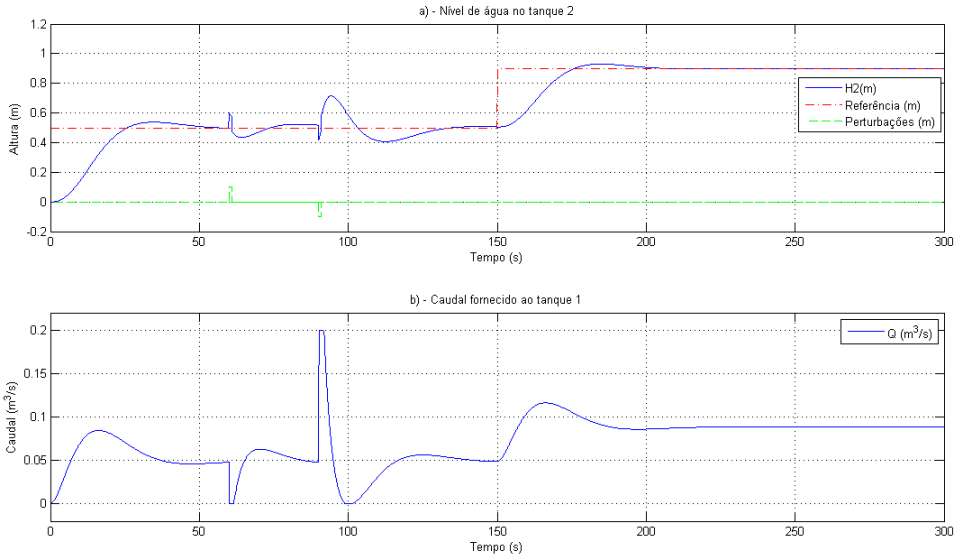

Figura 4.49: Resposta do sistema de dois tanques acoplados com servo controlo.

Capítulo 5

Conclusões Gerais e Propostas de Trabalho Futuro

Como conclusões gerais deste trabalho pode afirmar-se que o principal objectivo foi cumprido, ou seja, conseguiu desenvolver-se um laboratório virtual de sistemas de controlo com diferentes casos de estudo usando a *Virtual Reality Toolbox* do software Matlab®, animando os mesmos segundo os modelos matemáticos obtidos. Outros objectivos, como o projecto dos controladores para cada caso de controlo foi também desenvolvido com relativo sucesso, dado que para alguns casos se obtiveram valores de parâmetros de difícil realização prática e consideraram-se desprezáveis algumas não linearidades. Contudo, as especificações dinâmicas definidas foram sempre cumpridas.

Relativamente a propostas de trabalho a desenvolver no futuro, existem algumas sugestões com especial interesse em evoluir. Entre estas destaco a implementação de uma aplicação de software independente da aplicação Matlab® e das suas *toolboxes (stand-alone application),* que permitisse realizar o estudo dos vários sistemas de controlo implementados em 3D a partir de uma interface gráfica de utilizador, ao invés da execução de *scripts*. Esta solução poderia partir da geração de código C dos modelos criados em Simulink®, usando também ferramentas Matlab® como Real-Time Workshop® ou Matlab® Compiler™. O *design* da interface gráfica de utilizador poderia ser implementado recorrendo à ferramenta Guide (também do Matlab®), ou através de outra linguagem de programação orientada a objectos (por exemplo Visual C ou Java).

Outra sugestão de trabalho prender-se-ia com o desenvolvimento de outros casos de estudo que fizessem uso de técnicas de controlo moderno ou técnicas de controlo clássico que não tenham sido utilizadas, como sejam o projecto de controladores usando as regras de Ziegler-Nichols ou Chien-Hrones-Reswick.

Adicionalmente, seria também interessante colocar estes modelos 3D num sevidor de modo a possibilitar a simulação dos mesmos remotamente. A implementação desta sugestão passaria por configurar os ambientes virtuais implementados, os blocos da *Virtual Reality Toolbox*, bem como os modelos Simulink® associados, de modo a estes estarem acessíveis via *web*. A execução destes modelos remotamente requeria apenas que o utilizador tivesse instalada uma versão compatível do Matlab® ou, caso o modelo virtual estivesse já animado, um visualizador de ficheiros *.wrl*.

Capítulo 6

Anexos

6.1 Anexo 1

```
%Projecto do compensador PI
clear all
close all
clc

open('Carro.mdl')
%Dados Sistema
tsim=60; %Tempo de simulação em segundos
M=700; %Massa do veículo em Kg
f=50; %força de atrito em N.m/s

wrl=vrworld('carnew.wrl');
x=nodes(wrl,'-full');
posini=wrl.Viper.translation;
campos=wrl.MovCamPar.translation;
campos1=wrl.MovCam.translation;
textpos=wrl.Textpos.translation;

%Especificações para o projecto do compensador PI
PO=0.01; %Factor de sobre-elevação
tr=10; %Tempo de subida
ess=0.02; %Erro em regime estacionário
qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
wn=(qsi*2.5+0.8)/tr;

%Valores de teste com compensador P
% Kp=1*(f-ess*f)/ess;
% Ki=0;

%Projecto de valores compensador PI
Ki=0.442*wn^2*M;
Kp=0.74*(2*qsi*wn*M-f);

vrview('carnew.wrl')
sim('Carro.mdl',tsim)

t=time.signals.values; %tempo de simulação
force=F.signals.values; %força gerada pelo motor
```

```
39 v=v_er.signals.values(:,2); %velocidade atingida pelo veículo
40
41 %Gráficos dos sinais do sistema
42 figure
43 subplot(2,1,1)
44 wep1=plot(t,v,'-b',t,vref.signals.values,'--r');
45 set(wep1,'LineWidth',1)
46 grid on
47 title('a)');
48 legend('Velocidade(m/s)', 'Velocidade de Referência (m/s)');
49 axis([0 tsim min(v) max(v)+1])
50 xlabel('Tempo (s)')
51 ylabel('Velocidade(m/s)')
52 subplot(2,1,2)
53 wep3=plot(t,force,'-b');
54 set(wep3,'LineWidth',1)
55 grid on
56 title('b)')
57 legend('Força (N)');
58 axis auto
59 xlabel('Tempo (s)')
60 ylabel('Força (N)')
61
62 %%
63 %Compensador Atraso
64 clc
65 open('Carro_At.mdl')
66 wrl1=vrworld('carnew.wrl');
67 reload(wrl1)
68
69 x=nodes(wrl1,'-full');
70 posini=wrl1.Viper.translation;
71 campos=wrl1.MovCamPar.translation;
72 campos1=wrl1.MovCam.translation;
73 textpos=wrl1.Textpos.translation;
74
75 %Projecto compensador cancelamento/implantação de pólo
76 lmb=0.1; %factor empírico
77 den=[M f]; %denomidador da função de transferência do carro
78 a=abs(roots(den));%pólo dominante do sistema não compensado
79 kat1=(wn^2*M-lmb*a*f)/a; %cálculo a partir de tr
80 kat2=2*qsi*wn*M-f-lmb*a*M; %cálculo a partir de PO e tr
81 kat3=((lmb*a*f)/ess - lmb*f*a)/a; %cálculo a partir de ess
82 kat=0.773*kat2;
83
84 vrview('carnew.wrl')
85 sim('Carro_At.mdl',tsim)
86
87 t=time.signals.values; %tempo de simulação
88 force2=F_cat.signals.values; %força gerada pelo motor
89 v2=v_er_cat.signals.values(:,2); %velocidade atingida pelo veículo
90 %Gráficos dos sinais do sistema
91 figure
92 subplot(2,1,1)
93 wep1=plot(t,v2,'-b',t,vref.signals.values,'--r');
94 set(wep1,'LineWidth',1)
```

```
95  grid on
96  title('a) - Variável controlada (Velocidade)');
97  legend('Velocidade(m/s)', 'Velocidade de Referência (m/s)');
98  axis([0 tsim 0 (max(v2)+1)])
99  xlabel('Tempo (s)')
100 ylabel('Velocidade (m/s)')
101 subplot(2,1,2)
102 wep3=plot(t,force2,'-.k');
103 set(wep1,'LineWidth',1)
104 grid on
105 title('b) - Força no motor')
106 legend('F (N)');
107 axis auto
108 xlabel('Tempo (s)')
109 ylabel('Força (N)')
110 %Comparação do compensador PI com compensador atraso
111 %Executar depois de simular 'Carro.mdl'
112 figure
113 subplot(2,1,1)
114 wep7=plot(t(1:length(v),1),v,'-k',t(1:length(v),1),v2(1:length(v),1),'--r');
115 set(wep7,'LineWidth',1)
116 title('Controlo de velocidade: Controlador Atraso vs. Controlador PI')
117 legend('Controlador PI','Controlador Atraso')
118 grid on
119 axis auto
120 xlabel('Tempo (s)')
121 ylabel('Velocidade (m/s)')
122 subplot(2,1,2)
123 wep8=plot(t(1:length(v),1),force,'-k',t(1:length(v),1),force2(1:length(v),1),'--g');
124 set(wep8,'LineWidth',1)
125 title('Força no motor: Controlador Atraso vs. Controlador PI')
126 legend('Controlador PI','Controlador Atraso')
127 grid on
128 axis auto
129 xlabel('Tempo (s)')
130 ylabel('Força (N)')
```

6.2 Anexo 2

```
1  %Ficheiro DC_dadosKp.m
2  close all
3  clear all
4  clc
5
6  open('DC_roldana_P.mdl')
7  tsim=40; % tempo de execução da simulação
8  %Dados Motor DC
9  J1=0.01; %Inércia das "Roldana 1" (Kg.m^2)
10 Jm=0.01; %Inércia do motor (kg.m^2)
11 D=0.1; % Força de atrito no eixo de saída(N.m/rad.s^-1)
12 La=0.5; %Indutância da Armadura (H)
13 Ra=1; % Resistência da Armadura (ohm)
14 Km=0.01; % Constante do Binário produzido pelo motor
15 Kg=0.01; %Constante do cálculo da f.e.m.
16 %%%%%Valores do ambiente virtual
17 myworld=vrworld('DCmotor.wrl');
```

```
x=nodes(myworld,'-full');
r=(myworld.Rold1Rim.scale(1,1))/2; %Raio das "Roldanas"
campos=myworld.Camera_line.translation;
posini=myworld.Missile.translation;

a2=1;
a1=((J1+Jm)*Ra+D*La)/((J1+Jm)*La);
a0=(D*Ra+Km*Kg)/((J1+Jm)*La);
b=Km/((J1+Jm)*La);

%Especificações de dinâmica
PO=0.1; %Factor de sobre-elevação
tss=10; %tempo de estabelecimento
ess=0.05; %Erro em regime estacionário
qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
wn=4/(qsi*tss);

%Projecto compensador Proporcional
kp1=(wn^2*(Jm+J1)*La-Km*Kg-D*Ra)/Km; %kp a partir do tss
kp2=(D*Ra+Km*Kg-ess*D*Ra-ess*Km*Kg)/(ess*Km); %kp calculado a partir de ess
Kp=max(kp1,kp2);
Kd=0;
Ki=0;

%%%%Visualização da simulação
vrview('DCmotor.wrl')
sim('DC_roldana_P.mdl',tsim)

%Resultados com compensador
t=VA_E.time;
w=VA_E.signals.values(:,1);
ref=VA_E.signals.values(:,2);
Vas=Va.signals.values;
figure
subplot(2,1,1)
wep1=plot(t,w,'-b',t,ref,'--r');
set(wep1,'LineWidth',1)
grid on
title('a) - Variável controlada: Velocidade angular');
legend('Velocidade Angular(rad/s)', 'Referência (rad/s)');
axis([0 t min(w)-0.01 max(w)+0.01])
xlabel('Tempo (s)')
ylabel('W (rad/s)')
subplot(2,1,2)
wep2=plot(t,Vas,'-b');
set(wep2,'LineWidth',1)
title('b) - Tensão na armadura do motor DC');
grid on
legend('Tensão Va (V)');
axis tight
xlabel('Tempo (s)')
ylabel('Va (V)')

%Compensador PI
clc
open('DC_roldana_PID.mdl')
```

```
74
75 %%%%%%Ambiente Virtual
76 myworld2=vrworld('DCmotor.wrl');
77 reload(myworld2);
78 x=nodes(myworld2,'-full');
79 posini=myworld2.Missile.translation;
80 posini2=myworld2.Missile2.translation;
81 posini3=myworld2.Missile3.translation;
82 r=(myworld2.Rold1Rim.scale(1,1))/2; %Raio das "Roldanas"
83 posang1=myworld2.Rold1Rim.rotation;
84 rot=[0 0 -1];
85
86 %Projecto do compensador PI
87 beta=(a1-2*qsi*wn);
88 Kpb=2*qsi*wn*beta/b;
89 Kib=a0*beta/b;
90 ki=0.45;
91 Ki=ki*Kib;
92 kp=4.9;
93 Kp=kp*Kpb;
94 Kd=0;
95
96 %%%%Visualização da simulação
97 vrview('DCmotor.wrl')
98 sim('DC_roldana_PID.mdl',tsim)
99
100 %Dados da simulação com PI
101 %Parâmetros do regime transitório e estacionário no Sisotool
102 num=[b];
103 den=[a2 a1 a0];
104 G=tf(num,den);
105 s=tf('s');
106 Gc=(Ki/s+Kp);
107 sisotool(G,Gc)
108
109 %%%%%%Gráficos
110 tpi=W_E_PID.time;
111 w2=W_E_PID.signals.values(:,1);
112 e2=W_E_PID.signals.values(:,2);
113 Vas2=Va_PID.signals.values;
114 figure
115 subplot(2,1,1)
116 wep3=plot(tpi,w2,'-b',tpi,e2,'--r');
117 set(wep3,'LineWidth',1)
118 grid on
119 title('a)- Velocidade angular');
120 legend('Velocidade angular(rad/s)', 'Referência (rad/s)');
121 xlabel('Tempo (s)')
122 ylabel('w (rad/s)')
123 subplot(2,1,2)
124 wep4=plot(tpi,Vas2,'-b');
125 set(wep4,'LineWidth',1)
126 title('b) - Tensão Va na armadura do motor DC');
127 grid on
128 legend('Tensão Va (V)');
129 xlabel('Tempo (s)')
```

```
130 ylabel('Va (V)')
```

6.3 Anexo 3

```
1  clc
2  close all
3
4  open('DC_roldana_avstd.mdl')
5  %Dados Motor DC
6  J1=0.01; %Inércia das "Roldana 1" (Kg.m^2)
7  Jm=0.01; %Inércia do motor (kg.m^2)
8  D=0.1; % Força de atrito no eixo de saída(N.m/rad.s^-1)
9  La=0.5; %Indutância da Armadura (H)
10 Ra=1; % Resistência da Armadura (ohm)
11 Km=0.01; % Constante do Binário produzido pelo motor
12 Kg=0.01; %Constante do cálculo da f.e.m.
13
14 %Ambiente Virtual
15 myworld=vrworld('DCmotor.wrl');
16 x=nodes(myworld,'-full');
17 posini=myworld.Missile.translation;
18 posini2=myworld.Missile2.translation;
19 posini3=myworld.Missile3.translation;
20 r=(myworld.Rold1Rim.scale(1,1))/2; %Raio das "Roldanas"
21 posang1=myworld.Rold1Rim.rotation;
22 rot=[0 0 -1];
23
24 %Coeficientes
25 a2=1;
26 a1=((J1+Jm)*Ra+D*La)/((J1+Jm)*La);
27 a0=(D*Ra+Km*Kg)/((J1+Jm)*La);
28 b=Km/((J1+Jm)*La);
29
30 %Especificações de dinâmica
31 %Executar antes de simular
32 PO=0.1;
33 tss=10; %tempo de estabelecimento
34 ess=0.1;
35 qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
36 wn=4/(qsi*tss);
37 pol2=[1 2*qsi*wn wn^2];
38 roots(pol2)
39 tsim=40;
40
41 %Projecto do compensador atraso
42 num=[b];
43 den=[a2 a1 a0];
44 G=tf(num,den);
45 polos_sys=roots(den);
46 lmb=0.1; %factor empírico
47 a=abs(max(polos_sys));
48 s=tf('s');
49 Gc=(s+a)/(s+lmb*a);%
50 kav=1;
51 Gol=G*Gc;
52 sisotool(Gol)
```

```
53  kav=30;
54
55  vrview('DCmotor.wrl')
56  sim('DC_roldana_avstd.mdl',tsim)
57
58  t2=W_E_avstd.time;
59  wavstd=W_E_avstd.signals.values(:,1);
60  ewavstd=W_E_avstd.signals.values(:,2);
61  Vas_avstd=Va_avstd.signals.values;
62  figure
63  subplot(2,1,1)
64  wep1=plot(t2,wavstd,'-b',t2,ewavstd,'--r');
65  set(wep1,'LineWidth',1.2)
66  % axis([0 t min(wavstd)-0.01 max(wavstd)+0.01])
67  grid on
68  title('a) - Velocidade Angular');
69  legend('w (rad/s)', 'Referência (rad/s)');
70  xlabel('Tempo (s)')
71  ylabel('Velocidade Angular (rad/s)')
72  subplot(2,1,2)
73  wep2=plot(t2,Vas_avstd,'-b');
74  set(wep2,'LineWidth',1.2)
75  axis auto
76  title('b) - Tensão na armadura do motor DC');
77  grid on
78  legend('Va (V)');
79  xlabel('Tempo (s)')
80  ylabel('Tensão (V)')
81
82  %Comparação PI vs. atraso vs avanco
83  figure
84  subplot(2,1,1)
85  weppi=plot(tpi,w2,'-b')
86  set(weppi,'LineWidth',1.5)
87  hold on
88  wepcp=plot(t2,wavstd,'--r')
89  set(wepcp,'LineWidth',1.5)
90  hold on
91  wepav=plot(t,w,'-.g')
92  set(wepav,'LineWidth',1.5)
93  hold off
94  xlabel('Tempo (s)')
95  ylabel('Velocidade Angular (rad/s)')
96  title('a) - Velocidade Angular com: Comp. PI vs. Comp. Cancel/Implantação vs. Comp. A
97  legend('PI','Comp. Cancel/Implantação','Comp. Avanço','Referência')
98  grid on
99  axis([0 t2 min(wavstd)-0.01 max(wavstd)+0.01])
100 xlabel('Tempo (s)')
101 ylabel('Velocidade Angular (rad/s)')
102 subplot(2,1,2)
103 Vapi=plot(tpi,Vas2,'-b')
104 set(Vapi,'LineWidth',1.5)
105 hold on
106 Vacp=plot(t2,Vas_avstd,'--r')
107 axis manual
108 set(Vacp,'LineWidth',1.5)
```

```
hold on
Vaav=plot(t,Vas,'-k')
set(Vaav,'LineWidth',1.5)
hold on
title('b) - Tensão na armadura com: PI vs. Comp. Cancel/Implantação vs. Comp. Avanço')
xlabel('Tempo (s)')
ylabel('Tensão Va (V)')
legend('PI','Comp. Cancel/Implantação','Comp Avanço')
```

6.4 Anexo 4

```
close all
clear all
clc

open('DC_roldana_avanco.mdl')
%Dados Motor DC
J1=0.01; %Inércia das "Roldana 1" (Kg.m^2)
Jm=0.01; %Inércia do motor (kg.m^2)
D=0.1; % Força de atrito no eixo de saída(N.m/rad.s^-1)
La=0.5; %Indutância da Armadura (H)
Ra=1; % Resistência da Armadura (ohm)
Km=0.01; % Constante do Binário produzido pelo motor
Kg=0.01; %Constante do cálculo da f.e.m.
r=0.5; %Raio das "Roldanas"

a2=1;
a1=((J1+Jm)*Ra+D*La)/((J1+Jm)*La);
a0=(D*Ra+Km*Kg)/((J1+Jm)*La);
b=Km/((J1+Jm)*La);

%Especificações de dinâmica
PO=0.1; %factor de sobre-elevação
tss=10; %tempo de estabelecimento
ess=0.1; %erro em regime estacionário
qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
wn=4/(qsi*tss);
tsim=40;

lmb=10;
Kpess=a0*(1-ess)/ess; %kp calculado a partir de ess
Kpwn=(wn^2-a0)/b;
Kdm=(2*qsi*wn-a1)/b;
Ki=0;
Kp=max(Kpess,Kpwn);
kd=14;
Kd=kd*max(Kdm,1);
% a partir das especificações
s=tf('s');
num=b;
den=[a2 a1 a0];
G=tf(num,den);
Gcalt=Kd*(s+Kp/Kd)/(s+lmb*Kp/Kd);
Gnovo=G*Gcalt;
sisotool(Gnovo)
kav=135;
```

```
46
47 vrview('DCmotor.wrl')
48 sim('DC_roldana_avanco.mdl',tsim)
49 %gráficos
50 t=VA_E.time;
51 w=VA_E.signals.values(:,1);
52 e=VA_E.signals.values(:,2);
53 Vas=Va.signals.values;
54 figure
55 subplot(2,1,1)
56 wep1=plot(t,w,'-b',t,e,'--r');
57 set(wep1,'LineWidth',1.2)
58 % axis([0 t min(w)-0.01 max(w)+0.01])
59 grid on
60 title('a) - Velocidade Angular');
61 legend('w (rad/s)', 'Referência (rad/s)');
62 xlabel('Tempo (s)')
63 ylabel('Velocidade Angular (rad/s)')
64 subplot(2,1,2)
65 wep2=plot(t,Vas,'-b');
66 set(wep2,'LineWidth',1.2)
67 axis tight
68 title('b) - Tensão na armadura do motor DC');
69 grid on
70 legend('Va (V)');
71 xlabel('Tempo (s)')
72 ylabel('Tensão (V)')
```

6.5 Anexo 5

```
1 close all
2 clear all
3 clc
4
5 open('TF_modeltank.mdl')
6 %Dados do Sistema
7 ro=1000; %Densidade (kg/m^3)
8 g=9.8; %Aceleração da gravidade (m/s^2)
9 A1=3; %Área de secção do tanque 1 real (m^2)
10 A2=3; %Área de secção do tanque 2 real (m^2)
11 R1=10e+1;
12 R2=1000e+2;
13 tsim=300;
14
15 %Constantes Ambiente virtual
16 mv=vrworld('watertanks.wrl');
17 x=nodes(mv,'-full');
18 water1pos=mv.WaterTank1.translation;
19 water2pos=mv.WaterTank2.translation;
20 R1sat=mv.R1.scale(1,2);
21 wp1=mv.waterP1.translation;
22 wp2=mv.waterP2.translation;
23 R2sat=mv.R2.scale(1,2);
24 caudalpos=mv.CaudalQ.translation;
25 caudalxzmax=[0.1 0.1];
26 % hr=mv.Tank1.scale(1,2);%Altura virtual dos tanques
```

```
hr=1;

%Coeficientes para cálculo da função de transferência
b=1/A1;
a0=ro*g/(A1*R1);
a1=ro*g/(A2*R1);
a2=ro*g*(R1+R2)/(A2*R1*R2);

%Especificações
PO=0.085;
tss=50; %tempo de estabelecimento
ess=0.1;
qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
wn=4/(qsi*tss);

%Cálculo do compensador Proporcional
num=[b*a1];
den=[1 (a0+a2) (a0*a2-a0*a1)];
G2=tf(num,den);
sisotool(G2)
Kp=25;
Ki=0;
Kd=0;

% vrview('watertanks.wrl')
sim('TF_modeltank.mdl',tsim)

%Gráficos
t=H2.time;
h2=H2.signals.values;
h1=H1.signals.values;
caudal=Qu.signals.values;
ref=Erro.signals.values;
ruido=Ruido.signals.values;
figure
subplot(2,1,1)
wep1=plot(t,h2,'-b',t,ref,'--r',t,ruido(:,1),'--g',t,ruido(:,2),'--g');
set(wep1,'LineWidth',1.2)
axis auto
grid on
title('a) - Nível de água no tanque 2');
legend('H2(m)', 'Referência (m)','Perturbação (m)');
xlabel('Tempo (s)')
ylabel('Altura (m)')
subplot(2,1,2)
wep2=plot(t,caudal,'-b');
set(wep2,'LineWidth',1.2)
axis([0 t(end) (min(caudal)-0.02) (max(caudal)+0.02)])
title('b) - Caudal fornecido ao tanque 1');
grid on
legend('Q (m^3/s)');
xlabel('Tempo (s)')
ylabel('Caudal (m^3/s)')
%%
%Compensador PI
beta=5.68*(a0+a2)/2*qsi*wn;
```

```
Kp=20*(2*qsi*wn*beta+wn^2-a0*a2+a0*a1)/(a1*b);
Ki=0.265*wn^2*beta/(b*a1);
Kd=0;

% vrview('watertanks.wrl')
sim('TF_modeltank.mdl',tsim)

clc
close all
%Gráficos
t2=H2.time;
h22=H2.signals.values;
h12=H1.signals.values;
caudal2=Qu.signals.values;
e2=Erro.signals.values;
ruido2=Ruido.signals.values;
figure
subplot(2,1,1)
wep1=plot(t2,h22,'-b', t2,e2,'-.r',t2,ruido2(:,1),'--g',t2,ruido2(:,2),'--g');
set(wep1,'LineWidth',1.2)
axis auto
grid on
title('a) - Nível de água no tanque 2');
legend('H2(m)','Referência (m)','Perturbação (m)');
xlabel('Tempo (s)')
ylabel('Altura (m)')
subplot(2,1,2)
wep2=plot(t2,caudal2,'-b');
set(wep2,'LineWidth',1.2)
axis([0 t2(end) (min(caudal2)-0.02) (max(caudal2)+0.02)])
title('b) - Caudal fornecido ao tanque 1');
grid on
legend('Q (m^3/s)');
xlabel('Tempo (s)')
ylabel('Caudal (m^3/s)')
%%
%Comparação entre controlador P e PI
close all
clc
figure
subplot(2,1,1)
wep1=plot(t2,h22,'-b',t,h2,'--r',t,ref,'-.g',t2,ruido2(:,1),'--g',t2,ruido2(:,2),'--g
set(wep1,'LineWidth',1.2)
axis auto
grid on
title('a) - Nível de água no tanque 2');
legend('H2(m) - PI','H2(m) - P','Referência (m)','Perturbações (m)');
xlabel('Tempo (s)')
ylabel('Altura (m)')
subplot(2,1,2)
wep2=plot(t2,caudal2,'-b',t,caudal,'--r');
set(wep2,'LineWidth',1.2)
axis ([0 t2(end) (min(caudal2)-0.02) (max(caudal2)+0.02)])
title('b) - Caudal fornecido ao tanque 1');
grid on
legend('Q (m^3/s) - PI','Q (m^3/s) - P');
```

```
xlabel('Tempo (s)')
ylabel('Caudal (m^3/s)')
```

6.6 Anexo 6

```
% SERVOMECANISMO DO TIPO 0 ( SEM PÓLO NA ORIGEM)
clear all
close all
clc

open('SM_controller1.mdl')
tsim=300;
ro=1000; %Densidade (kg/m^3)
g=9.8; %Aceleração da gravidade (m/s^2)
A1=1; %Área de secção do tanque 1 real (m^2)
A2=1; %Área de secção do tanque 2 real (m^2)
R1=10e+1;
R2=1000e+2;

mv=vrworld('watertanks.WRL');
x=nodes(mv,'-full');
water1pos=mv.WaterTank1.translation;
water2pos=mv.WaterTank2.translation;
water1scale=mv.WaterTank1.scale;
water2scale=mv.WaterTank2.scale;
R1sat=mv.R1.scale(1,2);
wp1=mv.waterP1.translation;
wp2=mv.waterP2.translation;
R2sat=mv.R2.scale(1,2);
caudalpos=mv.CaudalQ.translation;
caudalxzmax=[0.1  0.1];

%Conversão de escalas
% hr=mv.Tank1.scale(1,2);
hr=1;

%Coeficientes para cálculo da função de transferência
b0=1/A1;
a0=ro*g/(A1*R1);
a1=ro*g/(A2*R1);
a2=ro*g*(R1+R2)/(A2*R1*R2);
b1=1/A2;

s = tf('s');
A=[-a0  a0;a1  -a2];
B=[b0;0];
D=[0  1];
E=[0];
sys1=ss(A,B,D,E)

%Analisar Observabilidade ( Sistema não compensado necessita ser completamente observa
Q = obsv(A,D);
no = length(A)-rank(Q) % Observável se no = 0
%Analisar Controlabilidade ( Sistema não compensado necessita ser completamente contr
S = ctrb(A,B);
nc = length(A)-rank(S) % Controlável se nc = 0
```

```
52 % As matrizes dos coeficientes e de entrada:
53 Z = zeros(length(A),1);
54 Atil = [A Z; -D 0];
55 Btil = [B;0];
56 Dtil = [D 0];
57
58 % Testar se o sistema correspondente ao erro é controlável em termos de estados.
59 Stil=ctrb(Atil,Btil);
60 Qtil=obsv(Atil,Dtil);
61 notil = length(Atil)-rank(Qtil) % Observável se notil = 0
62 nctil = length(Atil)-rank(Stil) % Controlável se nctil = 0
63 N = [A B; -D 0];
64 nn = length(A) + 1 - rank(N); % Controlável se nn = 0
65
66 % Pólos desejados do controlador que cumprem os requisitos + 1 colocado 4x mais à esqu
67 %Especificações
68 PO=0.085;
69 tss=50; %tempo de estabelecimento
70 ess=0.1;
71 qsi=sqrt((log (PO))^2)/sqrt((log (PO))^2 + pi^2);
72 wn=4/(qsi*tss);
73 poli=[1 2*qsi*wn wn^2];
74 polosw=roots(poli);
75 Pc =[polosw' min(real(polosw))*4];
76 % Matriz de ganho do sistema compensado
77 Ktil = place(Atil,Btil,Pc);
78 % Ganho do compensador integrador
79 ki = -Ktil(:,end);
80 K = Ktil(:,1:end-1);
81
82 % Equações Dinâmicas do Sistema compensado:
83 ACL = [A-B*K B*ki; -D 0];
84 BCL = [Z;1];
85 DCL = [D 0];
86 sys = ss(ACL,BCL,DCL,0);
87 G=tf(sys);
88 % step(sys)
89 % vrview('watertanks.WRL')
90 sim('SM_controller1.mdl',tsim)
91 %Gráficos
92 ts=H2.time;
93 h2s=H2.signals.values;
94 caudals=Qu.signals.values;
95 e=Erro.signals.values;
96 d1=D1.signals.values;
97 d2=D2.signals.values;
98 figure
99 subplot(2,1,1)
100 wep1=plot(ts,h2s,'-b',ts,e,'-.r',ts,d1,'--g',ts,d2,'--g',t,h2,'--k');
101 set(wep1,'LineWidth',1.2)
102 axis auto
103 grid on
104 title('a) - Nível de água no tanque 2');
105 legend('H2(m)','Referência (m)','Perturbações (m)');
106 xlabel('Tempo (s)')
107 ylabel('Altura (m)')
```

```
108 subplot(2,1,2)
109 wep2=plot(ts,caudals,'-b',t,caudal,'--r');
110 set(wep2,'LineWidth',1.2)
111 axis([0 t(end) min(caudals)-0.02 max(caudals)+0.02])
112 title('b) - Caudal fornecido ao tanque 1');
113 grid on
114 legend('Q (m^3/s)');
115 xlabel('Tempo (s)')
116 ylabel('Caudal (m^3/s)')
117
118 %%
119 % PI vs. Servo
120 figure
121 subplot(2,1,1)
122 gf1=plot(ts,h2s,'-b',ts,e,'-.r',t2,h22,'--k',ts,d1,'--g',ts,d2,'--g');
123 set(gf1,'LineWidth',1.5)
124 xlabel('Tempo (s)')
125 ylabel('Altura (m)')
126 axis auto
127 grid on
128 title('a) - Nível de água no tanque 2');
129 legend('H2 - Servo (m)','Referência (m)','H2 - PI (m)','Perturbações (m)');
130 subplot(2,1,2)
131 gf2=plot(ts,caudals,'-b',t2,caudal2,'--r');
132 set(gf2,'LineWidth',1.5)
133 xlabel('Tempo (s)')
134 ylabel('Caudal (m^3/s)')
135 axis([0 ts(end) min(caudals)-0.02 max(caudals)+0.02])
136 grid on
137 title('b) - Caudal fornecido ao tanque 1');
138 legend('Q - Servo (m^3/s)','Q - PI (m^3/s)');
```

Bibliografia

[1] URL http://www.xvrml.net/tutorial/images/rightHandRule.png.

[2] Apontamentos das aulas práticas de sistemas e controlo 1 (2007-2008).

[3] Apontamentos das aulas práticas de sistemas e controlo 2, 2008.

[4] How segways work, 2010. URL http://science.howstuffworks.com/ginger2.htm.

[5] Cambridge.University.Engineering.Dept, 2009. URL http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html.

[6] Team ECOSSE. The ecosse control hypercourse, . URL http://eweb.chemeng.ed.ac.uk/courses/control/course/map/index.html.

[7] Team ECOSSE. Hot water tank experiment, . URL http://eweb.chemeng.ed.ac.uk/courses/control/course/map/tank/index.html.

[8] Team ECOSSE. Level control experiment, . URL http://eweb.chemeng.ed.ac.uk/courses/control/course/map/level/question.html.

[9] Mark Pesce Gavin Bell, Anthony Parisi. The virtual reality modeling language, 2009. URL http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html.

[10] Petia Georgieva. Sistemas e controlo 2 - aulas práticas (2009/2010) trabalho prático 2, 2010.

[11] Graham Goodwin. Available packages vl-csd, 2009. URL http://www.virtual-laboratories.com/html/available_packages.htm.

[12] Graham Goodwin. Continuous caster - classical control design virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/caster_a.htm.

[13] Graham Goodwin. Electromechanical servomechanism - classical design virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/servo_simple.htm.

[14] Graham Goodwin. Electromechanical servomechanism - internal model control design virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/servo_imc.htm.

[15] Graham Goodwin. Fluid level dynamics and control - coupled tanks control virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/fluid_b.htm.

[16] Graham Goodwin. Rockets - dynamics virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/rockets_1.htm.

[17] Graham Goodwin. Rockets - controller design virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/rockets_2.htm.

[18] Graham Goodwin. Rolling mill - system modelling and classical control virtual laboratory, 2009. URL http://www.virtual-laboratories.com/html/rolling_a.htm.

[19] Graham Goodwin. Virtual laboratories for control system design, 2009. URL http://www.virtual-laboratories.com/.

[20] Humusoft and The Mathworks Inc. *Virtual Reality Toolbox - User's Guide - Version 3*, October 2004.

[21] Instituto Politécnico de Tomar IPT. 1º trabalho prático de controlo inteligente.

[22] J.L.Azevedo. Arquivo técnico micro-rato. URL `HTTP://MICRORATO.UA.PT/`.

[23] Y. Li K.H. Ang, G.C.Y. Chong. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13, 2005.

[24] Benjamin C. Kuo. *Automatic Control Systems*. Prentice Hall, seventh edition, 1995.

[25] L. Lemay. *3D Graphics & VRML 2*. Sams.net Publishing, first edition, 1996.

[26] The Mathworks. *MATLAB® 7 - Getting Started Guide*, .

[27] The Mathworks. *Simulink® 7 - Getting Started Guide*, .

[28] António Melo. *Sistemas e Controlo Cap. 11C*. 2008.

[29] Cleve Moler. The origins of matlab. URL `http://www.mathworks.com/company/newsletters/news_notes/clevescorner/dec04.html`.

[30] Regents of the University of Michigan. Carnegie mellon - control tutorials for matlab - pid design method for dc motor speed control. URL `http://www.engin.umich.edu/group/ctm/examples/motor/PID2.html`.

[31] Katsuhiko Ogata. *Modern Control Engineering*. 3rd edition, 1997.

[32] University.of.Bochum. Virtual control lab 3.1, 2002. URL `http://www.atp.ruhr-uni-bochum.de/VCLAB/`.