UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

ÁREA DE CONCENTRAÇÃO:

DEPARTAMENTO DE PROCESSOS QUÍMICOS

Identificação de sistemas "on-line", otimização e controle avançado com o filtro de Kalman extendido

Autor: Ir. Ramón Scheffer

Orientador: Prof. Dr. Rubens Maciel Filho

Dissertação de Tese de Doutorado apresentada à Faculdade de Engenharia Química como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Química.

Campinas - São Paulo

Janeiro 2006

FICHA CATALOGRÁFICA ELABORADA PELA

BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Sch24o

Scheffer, Ramón

Identificação de sistemas "on-line", otimização e controle avançado com o filtro de Kalman extendido. / Ramón Scheffer.--Campinas, SP: [s.n.], 2006.

Orientador: Rubens Maciel Filho

Tese (Doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Química.

1. Otimização matemática. 2. Controle de processos químicos. 3. Kalman, Filtragen. 4. Controle em tempo real. 5. Redes neurais (Computação). I. Maciel Filho, Rubens. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química. III. Título.

Titulo em Inglês: On-line system identification, advanced control and optimisation with the (Extended) Kalman filter

Palavras-chave em Inglês: Non-linear constrained optimisation, Advanced control, Kalman filter, Neural network

Área de concentração: Processos Químicos

Titulação: Doutor em Engenharia Química

Banca examinadora: Darcy Odloak, Wagner Caradori do Amaral, Valdir Apolinário de Freitas e Alexandre Tresmondi

Data da defesa: 16/01/2006

Dissertação de Tese de Doutorado defendida por Ramón Scheffer e aprovada em dia 16 de janeiro de 2006 pela banca examinadora constituída pelos doutores:

Dissertação de Tese de Doutorado defendida por Ramón Scheffer e aprovada em dia 16 de janeiro de 2006 pela banca examinadora constituída pelos doutores:

[illegible]

Prof. Dr. - Rubens Maciel Filho

[illegible]

Prof. Dr.Darcy Odloak

[illegible]

Prof. Dr. Wagner Caradori do Amaral

Apolina[illegible]

Dr. Valdir Apolinário de Freitas

[illegible]

Dr. Alexandre Tresmondi

200809059

Este exemplar corresponde à versão final da Tese de Doutorado em Engenharia Química.

[illegible]

Orientador: Prof. Dr. Rubens Maciel Filho

To my own beloved family, Priscila and my sons Tomás and Theo, to my family Jan and Ida, Mylou and Ilse & Olaf & Jelle, to the family I got as a present with my wife, Marco, Elza, Vanessa & Leandro and Dona Ruth, all for the close and long distance support, the love and companionship.

Agradecimentos

It has been seven years now living in Brazil, which would not be possible without a lot of support from a lot of people (living and finishing my thesis). First of all, the emotional support of my partner Priscila helping me to finish this work. My sons, Tomás en Theo for giving me the necessary distraction between work and thesis. My parents for calling me all the time from Holland, supporting me from the day I born by letting me make my choices in life, but always giving me some advice of want they think is right, and the phrase I think I heard most of them was: "want de laatste loodjes wegen het zwaarst, hé zoon".

To Professor Rubens Maciel Filho, which accepted me right away to work with him and enable me to come to Brazil. Of course his support and effort helped me a lot in studying the concepts in this thesis. His positive way of thinking, that makes things so much easier in live.

To my parents-in-law, Marco and Elza, which always supported me from the day that I arrived in Brasil, even supported me living in their house for some months. That says enough I think.

The friends I made in the laboratory LOPCA/LDPS/UNICAMP: Aline & Martin, Renata & Jair (Tubarão), Meleiro, Rubão, Luciana & Gerson, Marlus & Marilsa, Edwin, Lufer & Alexandra and sons, and others for their companionship, barbeques and making me feel at home.

To Wilson Martins for giving me the chance to start a professional life during my thesis, enabling to support my family. His guidance and life's advices and how to combine a good cultural life with work and study.

To those people from Rhodia, which became friends instead of just colleagues, one of the really nice things of Brazil. Ana Lucia, Jair (Tubarão), Wilson, Sergião, Fernandinho, Rogerio, Rose, Danilo, Virginia, Valdir, Aimar, Lionel and Eric, for all the happy hours and barbeques.

To Rhodia Brasil Ltda. for letting me use some of their industrial data in this study and therefore have an industrial validation of the ideas defended in this thesis.

To CAPES for giving me a scholarship before starting my professional life.

Epigrafe

In Holland it is common to put some philosophical phrases, so here they are:

A computer calculus can give you the wrong answer in the most precise way.

It is difficult to accept, how you calculate a stochastic number with the most deterministic equipment (computer) ever build by human kind

Those who want everybody to be treated equal, treat people less equal than those who accept people’s differences.

Are we in a big “condominium”? A question asked by my son Tomás on visiting Holland this year, as he could play on the street without us being afraid. In fact, Europe is becoming one big condominium…

If you want to see differences in cultures, travel fast, as global franchising is on its way.

Resumo

O processamento dos dados e a otimização dos processos químicos em tempo real ficarão mais importante com a competição crescente entres os produtores. Vários itens devem ser considerados para possibilitar a otimização em tempo real, como a medição, a confiança da medida e a predição do comportamento do processo.

Neste trabalho considera-se vários aspectos de um esquema de controle avançado destes, quais são a monitorização de medida, identificação de sistema não linear e em tempo real (redes neuronais recorrentes) e otimização não linear com restrições. Um requisito é que este sistema é capaz de funcionar em condições severas com ruído da medição, perturbações não medidas e mudanças de processo, como a desativação de um catalisador.

Todas estas ferramentas foram desenvolvidas na linguagem de programação FORTRAN e são disponíveis no laboratório LOPCA/UNICAMP. Utilizaram-se modelos validados para simular os processos, porém em alguns casos utilizaram-se dados industriais e dados de planta piloto para estudar os algoritmos desenvolvidos nesta tese.

O ruído Gaussiano fracionário (fGn = fractional Gaussian noise) e o movimento Browniano fracionário (fBm = fractional Brownian motion) foram considerados de ser modelos adequados para monitorização de medida e foram aplicados nos dados de um piloto de um reator air-lift, cujo sinal de pressão demonstra um comportamento complexo e não branco (não aleatório).

Demonstrou-se que o fGn descreve parcialmente os sinais da pressão e é capaz de prever os series temporais, porém, o parte que não era previsto bem pode ser previsto por um modelo (4,3) auto-regressivo e media móvel (ARMA = auto-regressive and moving average). Os modelos de fGn e fBm hão falta número de parâmetros ajustáveis e necessários para poderem ser utilizados em previsão de series temporais que tem uma função de auto-correlação de tipo senoidal. Portanto, recomenda-se o estudo da extensão do modelo ARMA que conhece-se por o modelo ARMA fracionário como algoritmo para monitorização da medida e por este via desenvolver uma ferramenta de diagnostica geral da confiança da medição.

O algoritmo de treinamento de redes neurais baseado no filtro de Kalman (MEKA) mostrou se bastante rápido para o ajuste dos parâmetros da rede neural recorrente em casos distantes, tanto em casos teóricos tanto em casos práticos de dados industriais. Alem disto, as características de generalização das redes neuronais treinados são melhores dos que as obtidas com os algoritmos comuns de treinamento de rede neural como standard backpropagation (com momentum).

Demonstrou-se com bastante sucesso que o filtro de Kalman pode ser utilizado em otimização com e sem restrições. A otimização sem restrições da função de Rosenbrock mostrou que o algoritmo pode ser muito rápido se a matriz de covarianca de ruído do processo é manipulada. A otimização com restrições demonstrou se em um escala grande de problemas de testes colecionados por Trvzka de Gouvêa e Odloak (), onde em quase todos os casos o ponto mínimo global foi encontrado. Alem disto utilizou se o algoritmo em um problema industrial que demonstrou que o custo computacional é alto demais ainda e que o algoritmo deveria ser modificado para ficar útil em aplicações reais.

Summary

In the continuing competition between it will be more and more necessary to optimize current chemical processes in real time. To be able to optimize a plant in real time, there have to be various aspects to be fulfilled, such as measurement, reliability of the measurement and prediction of the process behaviour.

In this work some of the aspects of such an advanced control are studied and are measurement monitoring, on-line non-linear system identification (recurrent neural networks) and constrained non-linear optimisation. It is wanted that this system can work under measurement noise, unmeasured disturbance and process changes such as a catalyst deactivation.

All these tools were developed in the FORTRAN programming language and are available at the laboratory LOPCA/UNICAMP. Validated models were used to simulate the processes, but in some cases real industrial and pilot-plant data were used to study the algorithms developed.

The fractional Gaussian noise (fGn) and fractional Brownian motion (fBm) were thought to be models suitable as measurement predictors, and applied to pilot plant data of an airlift reactor, whose pressure signal presents a complex non-white behaviour.

It was shown that the fGn does describe part of the measured signals and is able to do some prediction of the time series, but the other part could be explained well by a (4,3) Auto-Regressive and Moving Average (ARMA) model. It was noted that the fGn and fBm lack parameters to be adjusted and cannot be used for processes having a sinus type of auto-correlation function (ACF). Therefore an extension of ARMA models known as the fractional ARMA (FARMA) models can be used as a measurement monitoring tool, allowing the possibility to develop a general diagnostic tool.

It is shown a various cases (from theoretical to practical industrial data) that the MEKA Kalman filter algorithm is a quite fast training algorithm for recurrent neural network training, but especially results in better generalisation properties of the neural network trained than the other sequential training algorithms (standard backpropagation (with momentum)).

It was shown that the Kalman filter can be successfully used in unconstrained and constrained optimisation. The unconstrained optimisation of the Rosenbrock function demonstrates that a very fast optimisation can be obtained by manipulating the process noise covariance matrix. The applicability to constrained optimisation was shown in a large scope of different test problems and one real industrial problem.

Sumário

Nomenclatura

A	Matriz de transição do estado
B	Matriz de transição da entrada
B_H	Variável aleatório, realização do movimento Browniano fracionário com uma dependência H do longo prazo
C	Matriz de transição da observação
C	Matriz de correlacão
d	Valor desejado da observação
E(x)	Expectação do variável x
E()	Função de minimização da rede neural
$e_{k,j}$	Erro do neurônio j em camada k
f()	Função de transferência dos neurônios, para os redes neurais tipicamente uma função sigmoidal
f()	Função a ser minimalisado
f(x)	Função de densidade de probabilidade de um variável aleatório x
F(x)	Função de probabilidade acumulado da variável aleatório x
G	Matriz de transição do ruído de processo
g()	restrições de inigualdade
GEKF	Algoritmo de filtro de Kalman estendido Global em treinamento de redes neurais (Global Extended Kalman filter algorithm)
H	Parâmetro de Hurst
h()	restrições de igualdade
J	Função de custo
K	Matriz de filtro de Kalman que ajusta a variável de estado
L()	Função Langraniano

L_{max}	Função de verossimilhança maxima
m	Variável manipulado
MEKA	Algoritmo de múltiplos filtros de Kalman para treinamento de rede neural (Multiple Extended Kalman filter Algorithm)
n_a	Número de respostas possíveis de um evento A
n	Número total de respostas possíveis
$N(\mu,\sigma)$	Distribuição normal com a média μ e desvio padrão σ
N_k	Número de neurônios em camada k
P	Matriz de covariância do erro da variável do estado X
P(A)	Probabilidade do evento A
P(A/B)	Probabilidade do evento A dado B
$P(x \leq x_i)$	Probabilidade de uma variável aleatório e continuo
Q	Matriz de covariância do ruído de processo
R	Desvio do médio acumulado
R	Matriz de covariância do ruído de observação
R/S	Alcança ajustado escalado
S	Desvio padrão
SBP	algoritmo de retro-propagação (Standard Back-Propagation algorithm)
u	Variável de entrada
u	Multiplicador de Langrange das restrições de inigualdade
v	Variável aleatório do erro de processo
w	Variável aleatório do erro de observação
$w_{k,ji}$	Peso (parâmetro) da rede neural do neurônio i em camada k-1 para neurônio j em camada k
x	Variável de estado
$\boldsymbol{x}$	Variável aleatório x

y	Observação do estado x
$y_{k,j}$	Saída de neurônio j em camada k

GREGO

α	Parâmetro de Momentum do algoritmo SBP com momentum
δ	Gradiente de erro local
$\Gamma()$	Função de erro
γ	Função de covariância
η	Parâmetro de aprendizagem
ρ	Função de auto-correlação
τ	Janela de tempo
ω	Multiplicadores de Langrange das restrições de igualdade
ξ	Realização de serie temporal da dependência de longo prazo

Subscripts

k	Unidade discreta de tempo

Superscripts

-	Estimativa a priori
+	Estimativa a posteriori

1. Introdução

Hoje em dia, as empresas vendam os seus produtos no mundo inteiro e hão vários competidores (globais). Provavelmente, somente aquelas empresas que oferecem os seus produtos com a melhor qualidade através de um processo de produção mais econômica conseguirão sobreviver. Controle de alta qualidade será um possível fator chave para a manutenção da indústria química, e o sistema de controle deveria incluir idealmente os objetivos do gerenciamento ou "business", como por exemplo, a produção de dia. Além disto, um melhor controle de qualidade pode resultar em uma redução de custo também por uma formação menor de subprodutos.

Vários aspectos devem ser considerados em um sistema de controle de um processo real (Figura 1); a parte de planejamento da produção feito por um sistema ERP não é considerado neste trabalho, mas daria normalmente restrições econômicas como a vazão necessária para fazer a produção do dia.

O sistema de controle deve considerar outros aspectos, além de manter o processo no atual ponto de operação.

- O sistema de controle deve ser confiável com uma eficiência de operação (OEE = Overall Equipment Efficiency) > 95%
- O sistema de controle deve ser robusto aos erros de modelação e ao desvio permanente da medição
- O sistema de controle deve ser capaz de identificar e descartar medições errôneas

Enfatizou-se em alguns itens só deste esquema, e são: a estimação do modelo de processo ou maquina com modelos paramétricos ou redes neurais (identificação de sistema não linear), a ação de controle calculada por um algoritmo de controle preditivo e a validação da medição por determinar a confiança da medição. Um controle "dual" não foi implementado, pois este não há uma solução real em muitos casos. A separação de calculo de ação de controle e estimação de parâmetros de modelo é coerente com o conceito de controlador auto-ajustavel de Astrom and Wittenmark (1995) que resulta em uma solução viável e sub-otimal.

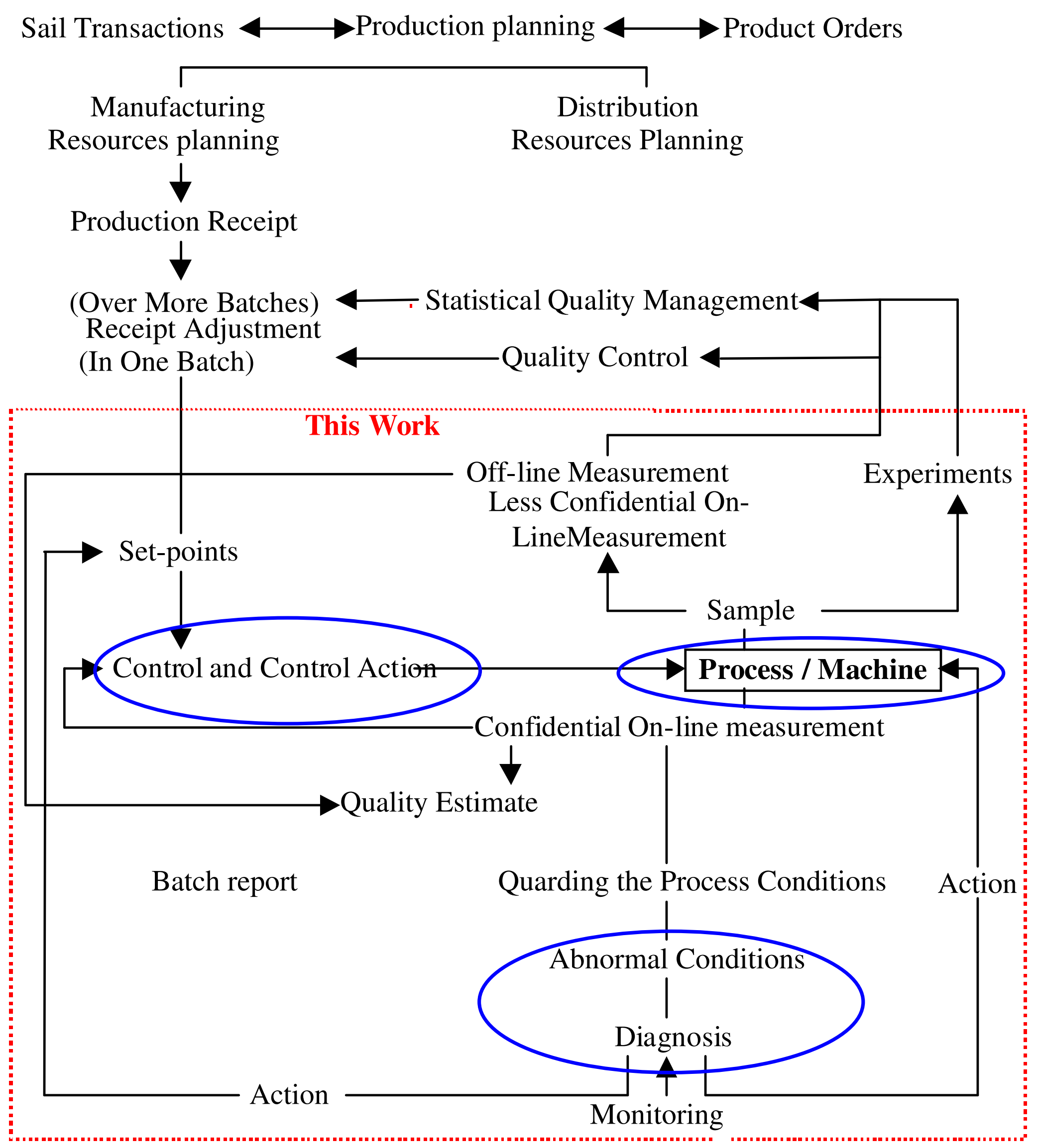

Sail Transactions
Production planning
Product Orders
Manufacturing Resources planning
Distribution Resources Planning
Production Receipt
(Over More Batches) Receipt Adjustment (In One Batch)
Statistical Quality Management
Quality Control
This Work
Off-line Measurement Less Confidential On-LineMeasurement
Experiments
Set-points
Sample
Control and Control Action
Process / Machine
Confidential On-line measurement
Quality Estimate
Batch report
Quarding the Process Conditions
Action
Abnormal Conditions
Diagnosis
Action
Monitoring

Figura 1: Um esquema simplificado das operações funcionais da indústria química (Rijnsdorp, 1993)

Estes três componentes formam exatamente a base de um controlador preditivo e não linear, que pode liderar com mudanças de processo através da possibilidade da adaptação do modelo, falha da medição e que implementa a melhor ação de controle possível considerando as interações das variáveis do processo.

Os principais objetivos desta tese são então:

- O desenvolvimento de um método não linear da identificação de sistema, cujos parâmetros são ajustados pelo filtro de Kalman em tempo real e é baseado em redes neurais recorrentes. O filtro de Kalman é um método de estimativa recursiva que pode resultar em um treinamento mais veloz e é capaz de funcionar em um ambiente com ruído de medida.
- Avaliação do ruído Gaussiano fracionário e do movimento Browniano fracionário como preditores da medida, portanto como monitores da medição.
- Desenvolvimento de um algoritmo de otimização não linear com e sem restrições baseado no filtro de Kalman estendido (EKF = extended Kalman filter) por ser um alternativo aos algoritmos de SQP e algoritmos genéticos. O algoritmo de EKF poderia ser utilizado em um algoritmo de controle preditivo ou utilizado na otimização do melhor ponto de operação do processo.

A identificação não linear de sistema aplicou-se ao processo de produção de Penicilina, que foi simulado através de um modelo detalhado do Rodrigues (1996). Este modelo foi validado por ele com dados experimentais do laboratório.

Um outro caso de estudo considerado é o processo da produção de fenol da Rhodia Ltda., qual consiste de uma cascada de quatro reatores air-lift. Este processo é equipado com medições em tempo real, que tem que ser desconectado de tempo em tempo para manutenção. Somente testes “off-line” foram feitas, mas simula a exploração em tempo real dos algoritmos desenvolvidos e como eles reagem com ruído da medida, falha de sensores e desvio da medida.

O ruído Gaussiano fracionário e movimento Browniano fracionário foram aplicados aos sinais de pressão do reator piloto de air-lift (Camerasaa et al., 2001), cujo sinal mostra um comportamento não aleatório em função das bolhas que passam pela coluna. Este piloto foi utilizado para estudos dos reatores industriais de “air-lift” de Rhodia Ltda.

O algoritmo EKF de otimização testou-se na função de Rosenbrock ou banana, que um problema “benchmark” em otimização sem restrições e é incluso no “Matlab optimisation’s toolbox”. Os problemas de teste para a otimização com restrições foram obtidos de Himmelblau (1998) e Trvska de Gouvêa and Odloak (1998) e formam um set de problemas bastante abrangente e de difícil solução.

Um caso de teste industrial foi a minimização das impurezas através um modelo detalhado dos reatores industriais de air-lift (Camarasa et al. 2001). Os resultados foram comparados com os do algoritmo comercial de SQP desenvolvido por Bartholows-Biggs da universidade de Hettersforth.

1.1. Organização de tese

Este tese não há uma organização comum, mas é um conjunto de vários artigos publicados em revistos e congressos internacionais, além da publicação de um capitulo de livro. Em cada capitulo é mencionado onde foi publicado, alem de haver um resumo em português para explicar os principais pontos do capitulo.

Em anexo 1, revê-se o filtro de Kalman e o filtro de Kalman estendido que é o fio vermelho deste tese. Uma introdução é dada sobre alguns aspectos como a teoria estatística e a teorema de Bayes, e a teoria de espaço de estado. O filtro de Kalman foi revisto em perspectiva determinística, que demonstra a derivação das equações do filtro, enquanto a perspectiva estatística demonstra a compreensão do algoritmo. Além disto, a visão estatística foi utilizado em desenvolvimentos recentes para melhorar o filtro de Kalman estendido.

Em capitulo 2 o filtro de Kalman é utilizado no treinamento de redes neuronais (recorrentes). Este trabalho foi publicado por uma apresentação **oral** no ESCAPE-10 (Conferencia internacional de Europian Society on Computers Applied to Process Engineering) e pelo capitulo de livro publicado pelo Imperial College em 2001: "Application of Neural Network and Other Learning Technologies in Process Engineering ", Ed. I.M.Mujtaba and M.A. Hussai.

Em capitulo 3, o filtro de Kalman é utilizado para treinamento de redes neurais, de qual a predição é utilizado por um outro filtro de Kalman para estimar a ação de controle.

Este esquema de controle é um controle não linear de variância mínima e foi publicado nos anais do sexto congresso mundial de engenharia química, Melbourne.

Em capitulo 4, o ruído Gaussiano fracionário e o movimento Browniano são aplicados para predizer o comportamento de sinais de pressão de reatores air lift. Este trabalhos foram publicados no CHEMICAL ENGINEERING SCIENCE, 56(2), 2001, pp. 707 – 711, “The fractional Brownian motion as a model for an industrial airlift reactor” e nos anais do 3th Europian Congress on Chemical Engineering.

O anexo 2 tratará a otimização do filtro de Kalman com e sem restrições. Este capitulo será submetido para a revista Computers and Chemical Engineering.

2. Modelos não lineares – Treinamento com o filtro de Kalman estendido

Resumo – Buscou-se um algoritmo mais adequado para o treinamento das redes neurais de tipo "feed-forward" e de tipos recorrente, especialmente para o uso deles em treinamento em tempo real. O algoritmo de treinamento é baseado no filtro de Kalman estendido, que é considerado em outras áreas técnicas como algoritmo de treinamento de redes neurais mais rápido e que converge para uma mínima global melhor do que os algoritmos normais como o algoritmo de retro-prorrogação (standard backpropagation). A vantagem da utilização do filtro de Kalman é sua abordagem recursiva e a possibilidade de trabalhar em um ambiente com ruídos de medição. As equações do filtro foram adaptados para admitir a adaptação ás mudanças de processo, o que faz o adequado para treinamento em tempo real. Isto é feito pela re-alimentação do erro de estimação na hora da actualização dinâmica da matriz de covariância do erro de processo, o que é visto como uma das contribuições deste trabalho na aplicação do filtro de Kalman em treinamento de redes neurais. A atualização da matriz de covariância do erro de ruído de processo é feita em uma maneira para garantir a estabilidade numérica do filtro. O algoritmo foi testado em varias aplicações como a identificação de um processo de penicilina e caso industriais (não publicado em artigo).

Demonstrou-se que o algoritmo de MEKA é bastante rápida, mas especialmente resulta em características de generalização melhor de uma rede neural comparado com os outros algoritmos de treinamento seqüencial. O algoritmo de backpropagation nunca resulta em um erro de treinamento e simulação baixa. Além disto a localização do erro não compromete o resultado. Estas habilidade de assegurar um bom treinamento com predição adequada dos dinâmicos do processo faz que o filtro de Kalman é um candidato preferencial para a identificação de sistema em um algoritmo de controle avançado.

Os trabalhos reproduzidos aqui são dois artigos, o primeiro foi publicado no **ESCAPE-10** como apresentação oral, o que resultou no convite para da 2ª publicação qual é **um capitulo no livro "Application of Neural Network and Other Learning Technologies in Process Engineering** ", Ed. I.M.Mujtaba and M.A. Hussai, publicado pelo Imperial College em 2001

TRAINING A RECURRENT NEURAL NETWORK BY THE EXTENDED KALMAN FILTER AS AN IDENTIFICATION TOOL

R. Scheffer and R. Maciel Filho

LOPCA/DPQ, Faculty of Chemical Engineering, State University of Campinas (UNICAMP), Cidade Universitária Zeferino Vaz, CP 6066, , Campinas – SP, Brazil, CEP 13081-970, e-mail: ramon@lopca.feq.unicamp.br

ABSTRACT

In this work we evaluate the behaviour of the extended Kalman filter as a recursive training algorithm for neural networks in real-time. It shows that the extended Kalman filter is a very potential candidate for on-line training of neural networks and as fast as the sequential backpropagation algorithm with momentum, but with better global convergence properties. It could not be shown that a recurrent neural network with external feedback is sufficient to give a dynamic representation of the process.

INTRODUCTION

Control of non-linear chemical processes relies on a good dynamical model, which usually is linear and has to be continuously updated to follow the process behaviour. In the past decade, this has lead to an increased interest for non-linear process control, which can be divided in optimisation and transformation methods. Only recently, artificial neural networks (ANN) attracted a great deal of attention, providing a simple way to describe non-linear processes at low computational cost. If trained by appropriate input/output data, an ANN is able to approximate any function mapping of the process input/output behaviour insides its training area. So, an ANN can be viewed as a black-box model of the process.

The classical approach to identify a non-linear dynamical process by an ANN, is to augment the number of inputs of the ANN by past values of the input data. For simple systems, which can be described by a one-dimensional state space system, this will result in a small and workable ANN. But this approach can result in too large ANN`s suitable for training more complicated systems, such as fixed-bed reactors, fluidised beds and bubble columns, and for systems with slow dynamics. Especially, when the process is subjected to environmental or internal changes, such as catalyst deactivation, the ANN should be dynamical and be trained in real time to be able to represent the process.

Therefore, the main objective of this work is the development of a dynamical ANN trained by a recursive algorithm as the Kalman filter, which can be used to identify a non-linear process on-line and afterwards to be used in a adaptive control scheme.

THEORY

The most popular ANN is the feedforwarded network trained by backpropagation of Rumelhart et al. (1986), which consists of an input layer, a hidden layer and an output layer. The output of a neuron j, y_j in the hidden or output layer k is calculated as a function of the outputs of the former layer as follows:

$$y_{k,j} = f\left(v_{k,j}\right) = f\left(\sum_{i=0}^{N_{k-1}} w_{k,ji} y_{k-1,i}\right) \tag{1}$$

where w_{ji} is the weight for the input x_i of neuron j and N_{k-1} is the number of inputs and the number of neurons in the former layer. The bias or treshold is defined by putting input $y_{k-1,0}$ equal to –1. k ranges from 1 to the number of chosen layers, where layer 0 is the input layer. The function *f(·)* is typically a sigmoidal or tangent hyperbolic function for a neuron in the hidden layer and linear for the output layer in case of function approximation. If the data is appropriately scaled, the latter can be a non-linear function also.

The ANN’s output and the desired output define an error, $e = d_j - y_j$, which can be propagated back through the system. The error for intermediate or hidden neurons is calculated by:

$$e_{k,j} = \sum_{j=1}^{p} w_{k+1,ji}\delta_{k+1,j} \tag{2}$$

$\delta_{k,j}$ is the local error gradient for neuron j in layer k and is calculated by:

$$\delta_{k,j} = \frac{df(v_{k,j})}{dv_{k,j}} e(k,j) \tag{3}$$

In case of the backpropagation algorithm, this leads to a parameter adjustment based on the steepest descent by:

$$w_{k,ji}(n+1) = w_{k,ji}(n) + \eta\partial_{k,j} y(k-1,i) \tag{4}$$

A faster convergence can be obtained by adding a momentum term (Rumerlhart et al., 1986), which makes the backpropagation algorithm more stable.

Optimization methods, as the conjugate gradients (Fletcher et al., 1964) and the method of Levenberg-Marquardt can be used to obtain a much faster convergence using second-order gradient information. These methods need a good estimate of the gradient, and therefore can only be used in a batch training mode, where an average gradient to the weights is calculated over the whole training set.

A dynamical ANN is known as a recurrent neural network (RNN), where some of the neurons in the layer k have a feedback connection with the neurons in layer l, where $l < k$. In this work was chosen only external feedback connections, which lead the outputs from the output layer back to the input layer. The advantage of this type of RNN is that during the training phase the target values instead of the RNN's outputs, can be fed to the input layer, which leads to a faster convergence. When the error of the output is small enough, the network outputs are fed to the input layer. In this way non-linear state-space approximations are trained by the network.

As for the ANN an error can be declared comparing the output of the RNN with the desired output (d), normally, a quadratic cost function of the error is declared, which can be minimised by the gradient methods mentioned above.

For on-line applications, the optimisation methods usually do not have a recursive calculation scheme and cannot be used, but the backpropagation algorithm is typically slowly and forgets the past data. A system identification method such as the Kalman filter could be used to update the networks parameters, which has the advantage to take into account the past data when it calculates a new optimal estimate with the new arrived data. The actualisation of the weight parameters by the Kalman filter is by definition of the following dynamical system:

$$
\begin{aligned}
& w_{k,ji}(n+1) = w_{k,ji}(n) + q_{k,ji}(n) \qquad i = 0 \ldots N_{k-1} \\
& d_{k,j}(n) = f\left(\sum_{i=0}^{N_{k-1}} y_{k-1,i}^{T}(n) w_{k,ji}(n) \right) + r_{k,j}(n)
\end{aligned} \tag{5}
$$

where $q_{k,ji}$ and $r_{k,j}$ are stochastic variables with a Normal random Gaussian distribution, N(0, Q) and N(0,R) respectively.

According to Shah e Palmieri (1990), the use of multiple extended Kalman filters (MEKA) can be used to train a feedforwarded neural network to obtain a much faster convergence than with the gradient methods. It has to be found out if this is valid for a recurrent network also. Puskorius et al. (1994) successfully implemented a decoupled Kalman filter algorithm training recurrent networks and were able to use these networks in various control problems.

An additional advantage is that MEKA can be used in a recursive scheme for process identification in a advanced control loop, without the disadvantages of the backpropagation algorithm.

Extended Kalman filter training algorithm

The training algorithm with the extended Kalman filter falls down into two parts, the dynamic actualisation and the actualisation of the observation. The dynamic actualisation is done by (5), where the estimates of q and r are 0. In fact, it is the same as the feedforwarded pass (1) of the backpropagation algorithm.

The only difference is that the Kalman filter algorithm needs to update the covariance matrix $P_{k,ji}$:

$$P_{k,jii}(n+1) = P_{k,jii}(n) + Q \tag{6}$$

Since the states of the dynamical system (5) are parameters, there would be in fact no process noise $q_{k,ji}$, but the addition of a very small process noise (in the order of 0.001) makes the filter more stable. A too large a process noise leads to oscillatory and stochastic behaviour.

The errors of the intermediate outputs are calculated by propagating back the error (eq 2 and 3). With the errors known for every neuron the multiple extended Kalman filters update the weight parameters by:

$$
\begin{aligned}
& w_{k,ji}(n+1) = w_{k,ji}(n) + K_{k,ji}(n) e_{k,j}(n) \qquad i = 0 \ldots N_{k-1} \\
& d_{k,j}(n) = \left[\frac{df(v_{k,j})}{dv_{k,j}} y_{k-1,0}(n) \quad \cdots \quad \frac{df(v_{k,j})}{dv_{k,j}} y_{k-1,N_{k-1}}(n) \right] \left[w_{k,j,0} \quad \cdots \quad w_{k,j,0} \right]^T = CW \\
& K_{k,ji} = {P_{k,jii} C^T} \Big/ {\left(C P_{k,jii} C^T + R \right)} \\
& P_{k,jii} = (I - KC) P (I - KC)^T + KRK^T
\end{aligned} \tag{7}
$$

The advantage of using a Kalman filter per neuron is that the product (CPC^T+R) becomes a scalar and therefore no matrix inversion is needed. The only disadvantage of the Kalman filter is the memory requirement for the process covariance matrix.

To ensure that the Kalman filter still reacts after some time , the matrixes Q or R have to be made a function of the error or the derivative of the error. This makes the system more sensible to new data.

The observation covariance matrix, R, has a large effect on the stability of the Kalman filter. With a too small value the filter diverges. R has to be increased when the number of neurons and or layers are increased, otherwise the filter becomes unstable.

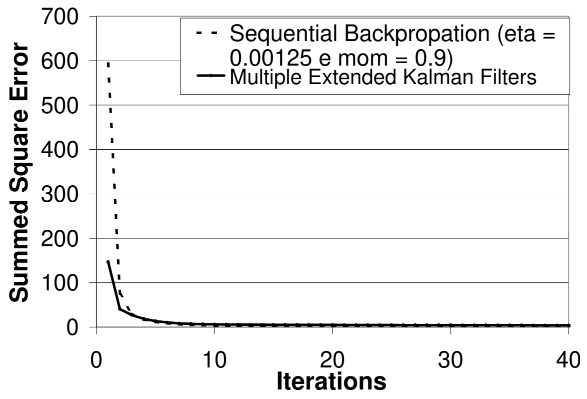
The matrix R was fixed and only made variable when the total summed square error of the outputs became lower than 5.0. If so, R was made equal to the total number of neurons multiplied with the total summed error.

Networks were trained with data obtained from simulation of a fed-batch penicillin process. The process model used was validated with experimental and pilot plant data. The

RNN`s were trained by the multiple extended Kalman filter, and by the different minimisation algorithms as conjugate gradients and Levenberg-Marquardt and the standard backpropagation algorithm with momentum in sequential and batch mode.

RESULTS

The ANN's trained with the different training algorithms were all able to give an accurate representation of the penicillin production process, when the target values of the outputs were not re-fed to the inputs. The convergence of the conjugate gradient method was much faster than the Levenberg-Marquardt algorithm, because the latter is much more dependent on the local gradient behaviour and the initial shoot (Figure 2). Both were much faster than the standard backpropagation algorithm with momentum. The ANN trained with the multiple extended Kalman filters was in the initial training phase faster than the sequential backpropagation algorithm with momentum, but became slower at the end phase (Figure 1).



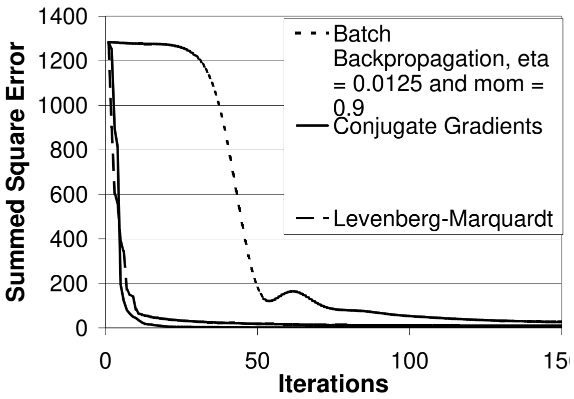


Figure 1: Summed square error for the sequential training algorithms for an ANN

Figure 2: Summed square error for the batch training algorithms for an ANN

This can mean that the chosen dependence of the Kalman filter to the observation covariance matrix should be changed to make the Kalman more sensible in the mid and end phases. But from a simple feedforwarded problem with the identification of a bioreactor, it showed that the generalisation performance of the multiple extended Kalman filters was two times better than with the backpropation algorithm (Table 1).

Table 1 Summed square errors of the test set for a simple feedforwarded network

Algorithm	Summed square error of the test set
Sequential Backpropagation (η= 0.00125 e α=0.9)	10.705
Multiple extended Kalman filters	*5.1281*
Batch Backpropagation (η= 0.0125 e α=0.9)	26.082
Gradientes Conjugados	*2.0119*
Levenberg-Marquardt	8.4759

In Figure 3, the error during the training of the RNN for the penicillin production process are shown, for when the *target values were re-fed*. It can be seen that the multiple extended Kalman filters converge in a low number of iteration, but stays on a higher error plane than the sequential Backpropagation algorithm. This does not mean automatically that the training by the sequential backpropagation algorithm is better that the extended Kalman filter algorithm, as could be seen from the identification of the feedforwarded network.

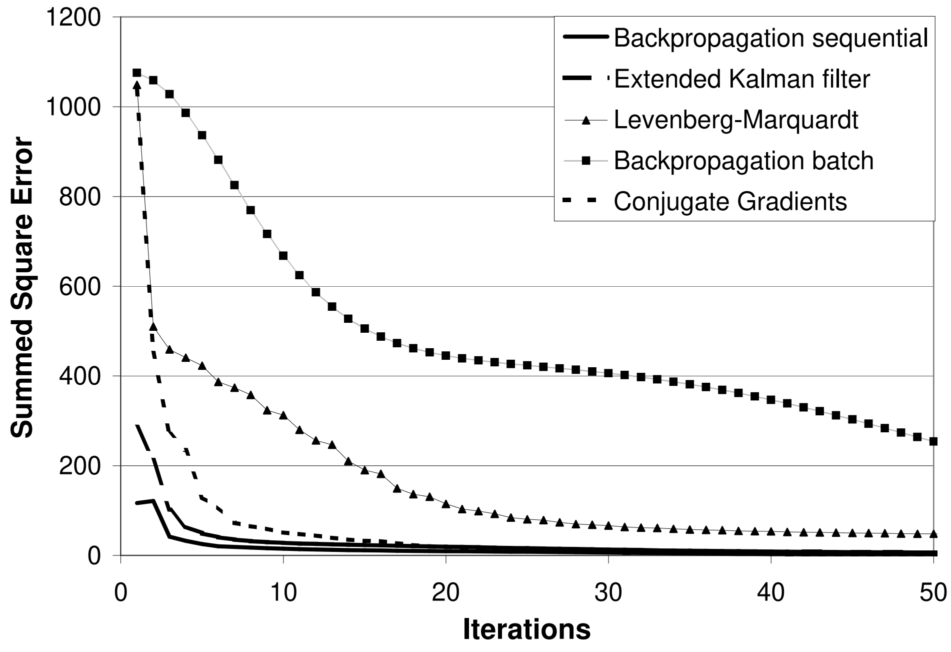


Figure 3: Summed square error for the different algorithms for identification of the penicillin production process by the RNN

When the outputs of the RNN were connected to the inputs this resulted in a high final error for the training set for all the training algorithms. This indicates that the RNN with only an external feedback is not enough to learn the dynamical behaviour of the penicillin production process and probably full recurrent network should be used.

CONCLUSIONS

It was shown that the multiple extended Kalman filters can be used as a training algorithm for neural networks in real-time. The extended Kalman filter converges faster and maintains better generalisation properties. It could not be shown that the RNN with external feedback can be used as an dynamical identification tool, and probably the RNN has to become fully recurrent to better describe non-linear chemical processes.

LITERATURE

Fletcher, R. and C. Reeves, *Function Minimization by Conjugate Gradients*, Computer Journal, 7, 149-154 (1964)

Puskorius, G.V. and L.A. Feldkamp, *Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks*, IEEE Transactions on Neural Networks, vol. 5, no. 2, 279-297 (1994)

Rumelhart, D.E. and J.L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, The MIT Press, Cambridge (1986)

Shah, S. e F. Palmieri, *MEKA – a fast, local algorihm for training feedforward neural networks*, San Francisco, CA, vol I, 226-231 (1990)

PROCESS IDENTIFICATION OF A FED-BATCH PENICILLIN PRODUCTION PROCESS - TRAINING WITH THE EXTENDED KALMAN FILTER

R. SCHEFFER, R. MACIEL FILHO
LOPCA/DPQ, Faculty of Chemical Engineering, State University of Campinas (UNICAMP), Cidade Universitária Zeferino Vaz,
CP 6066, , Campinas – SP, Brazil, CEP 13081-970,

Advcanced process control is typically based on a good process model. In this work a recurrent neural network is applied as a non-linear identification tool of a fed-batch penicillin process. The process is trained by a multiple-stream extended Kalman filter, which allows the process to be identified in real-time. It is shown that the fed batch process can be estimated very accurately by a recurrent network and that the extended Kalman filter is a very efficient and rapid training algorithm for non-linear process identification. The recurrent neural network could be trained to give an one-step ahead prediction for a sample-time of six minutes. This ensures that the neural network can be used as an identification model in a model predictive control loop for calculation of the optimal feeding strategy in real-time.

1. Introduction

Control of non-linear chemical processes relies on a good dynamical model, which usually is linear and has to be continuously updated to follow the process behaviour. In the past decade, this has lead to an increased interest for non-linear process control, which can be divided in optimisation and transformation methods. Only recently, artificial neural networks (ANN) attracted a great deal of attention, providing a simple way to describe non-linear processes at low computational cost. If trained by appropriate input/output data, an ANN is able to approximate any function mapping of the process input/output behaviour insides its training area. So, an ANN can be viewed as a black-box model of the process.

The classical approach to identify a non-linear dynamical process by an ANN, is to augment the number of inputs of the ANN by past values of the input data. For simple systems, which can be described by a one-dimensional state space system, this will result in a small and workable ANN. But this approach can result in too large ANN's suitable for training more complicated systems, such as fixed-bed reactors, fluidised beds and bubble columns, and for systems with slow dynamics. Especially, when the process is subjected to environmental or internal changes, such as catalyst deactivation, the ANN should be dynamical and be trained in real time to be able to represent the process.

Therefore, the main objective of this work is the development of a dynamical ANN trained by a recursive algorithm as the Kalman filter, which can be used to identify a non-linear process on-line and afterwards to be used in a adaptive control scheme. The Kalman filter´s parameters as the process and measurement noise covariance matrixes are made a function of the error to make the filter useful for on-line training.

2. Penicillin Production process

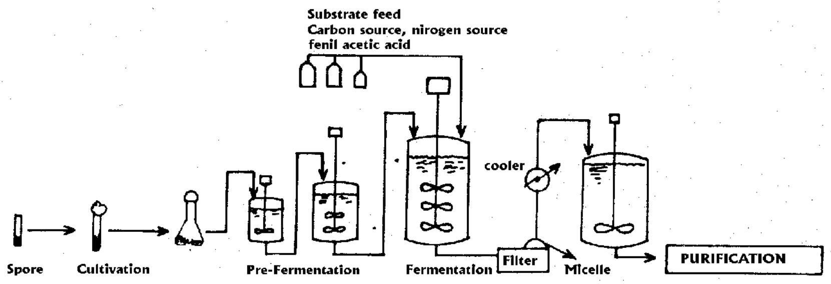


Figure 4: Fluxogram of the fed-batch production process (Crueger and Crueger, 1984)

Biochemical processes are mostly very sensitive processes and small changes in the process operation can lead to activity loss of the organism. Only basic process variables, such as the temperature, pressure and the pH are available at short measurement times. But variables as the bio-mass concentration or dissolved oxygen concentration are typically available at large measurement delays only.

To be able to control such a process a good process model is needed, but suffers from knowledge of the organism kinetics, which can depend, in a complicated way, on the process variables. Therefore such systems are ideal candidates to be identified by neural networks, which can map the non-linear system behaviour for relative large sampling intervals.

The penicillin production process can be divided into two phases, the growing phase and the production phase. The production phase is initiated by changing the feeding strategy. In the growing phase a large amount of sugar is added to enhance the growing of the organism. When the bio-mass concentration is high enough, the substrate concentration is lowered to encourage the penicillin production. To get a high end-penicillin concentration and a good final product quality the feeding strategy is essential during the production phase.

Rodrigues (1996) determined an optimal feeding strategy for this studied penicillin process off-line, but will be optimal only if the exact specifications can be followed. If the system is subjected to changes this strategy will become sub-optimal and there will be another strategy which will be optimal.

To determine this new optimal feeding strategy, the process model has to be updated to incorporate the changes which were not accounted for in the model. If the model is updated, it can be used in an optimisation criterion to calculate the new best control actions to be taken. A schematic of this type of control system is shown in Figure 5.

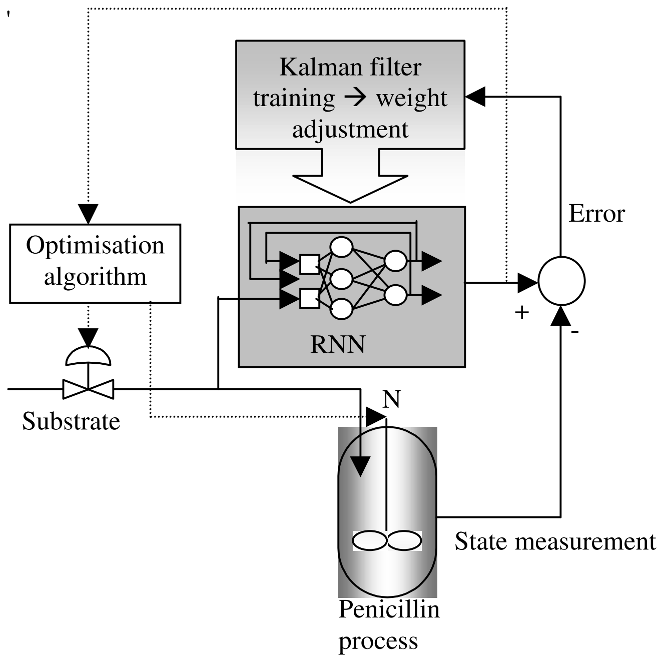


Figure 5: Schematic of the objected control scheme

The data of the penicillin process was obtained from a detailed model of Rodrigues (1996), which was validated with experimental data. Data of the penicillin process was collected at a sampling time of 6 minutes. The operating variables obtained were the substrate feed flow as the input variable and the bio-mass concentration, the substrate concentration, the penicillin concentration and the dissolved oxygen concentration in the reactor as the state variables. This data was learned to various recurrent neural networks of different sizes and architectures.

3. Training Neural Networks with the Extended Kalman Filter

The training of a neural network with the Kalman filter is done in conjunction with the back-propagation algorithm. The back-propagation pass is used to calculate the derivatives and the errors of every neuron in the network. The weight adjustment done with the Extended Kalman Filter, will give a much faster convergence than with the back-propagation method as it is a second order method based on the least square principal. But the storage and computational requirements may become too high as the network size is increased. In the next subsection a short review is given of the standard calculation of the common neural network and the way how the recurrent neural network is implemented. Afterwards, the differences are shown for applying the Kalman filter theory to the training of neural networks. For those readers unfamiliar with the Kalman filter theory, a derivation of the filter is shown in the appendix for a one-dimensional linear system to make the article self-containing.

2.1. Principle calculations of a neural network

The most popular ANN is the multiple layer perceptron trained by backpropagation of Rumelhart et al. (1986), which consists of an input layer, a number of hidden layers and an output layer. The output of a neuron j, y_j in the hidden or output layer k is calculated as a function of the outputs of the former layer as follows:

$$y_{k,j} = f\left(v_{k,j}\right) = f\left(\sum_{i=0}^{N_{k-1}} w_{k,ji}\, y_{k-1,i}\right) \tag{8}$$

where $w_{k,ji}$ is the weight for the connection of input y_i with neuron j in layer k and N_{k-1} is the number of inputs or the number of neurons in the former layer as the network is fully connected. The bias or treshold is defined by putting input $y_{k-1,0}$ equal to –1. k ranges from 0 to the number of chosen layers, where layer 0 is the input layer. The function *f(·)* is typically a sigmoidal or tangent hyperbolic function for a neuron in the hidden layer and linear for the output layer in case of function approximation. If the data is appropriately scaled, the latter can be a non-linear function also. A signal flow presentation is shown in Figure 6.

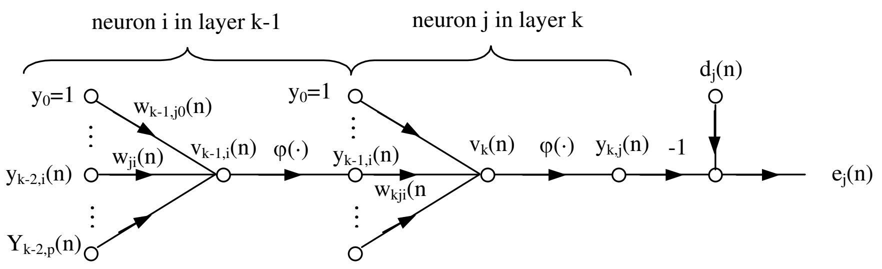


Figure 6: Signal flow diagram of neurons in subsequent layers in a neural network

The ANN's output and the desired output define an error, $e = d_j - y_j$, which can be propagated back through the system. The error for intermediate or hidden neurons is calculated by:

$$e_{k,j} = \sum_{j=1}^{p} w_{k+1,ji}\delta_{k+1,j} \tag{9}$$

$\delta_{k,j}$ is the local error gradient for neuron j in layer k and is calculated by:

$$\delta_{k,j} = \frac{df\left(v_{k,j}\right)}{dv_{k,j}} e\left(k,j\right) \tag{10}$$

In case of the backpropagation algorithm, this leads to a parameter adjustment based on the steepest descent by:

$$w_{k,ji}\left(n+1\right) = w_{k,ji}\left(n\right) + \eta\partial_{k,j}\,y\left(k-1,i\right) \tag{11}$$

A faster convergence can be obtained by adding a momentum term (Rumerlhart et al., 1986), which makes the backpropagation algorithm more stable because of the feedback of the last calculated parameter adjustment.
Optimization methods, as the conjugate gradients (Fletcher et al., 1964) and the method of Levenberg-Marquardt can be used to obtain a much faster convergence using second-order gradient information. The conjugate gradient formula of Polak-Ribiére was used as described by Haykin (1999). A line search is conducted in the calculated direction by a quadratic approximation. The Hessian matrix for the Levenberg-Marquardt method was approximated by the square of the Jacobian matrix, but gives satisfactory results.
These methods need a good estimate of the gradient, and therefore can only be used in a batch training mode, where an average gradient to the weights is calculated over the whole training set.

A dynamical ANN is known as a recurrent neural network (RNN), where some of the neurons in the layer k have a feedback connection with the neurons in layer l, where l < k. In this work was chosen only external feedback connections, which lead the outputs from the output layer back to the input layer. The advantage of this type of RNN is that during the training phase the target values instead of the RNN's outputs, can be fed to the input layer so-called teacher forcing, which leads to a faster convergence. When the error of the output is small enough, the network outputs are fed to the input layer. In this way non-linear state-space approximations are trained by the network.

If re-alimentation of the outputs is not sufficient, then more memory has to be build in the ANN. This can be done by applying a tap-delay filter of order q to the inputs and the re-fed outputs as demonstrated in Figure 7 for a RNN with one input and one output with a tap-delay filter of order 2 for both input and re-fed output. The present and past values of the input represent exogenous inputs, while the delayed values of the output form the regressive inputs of the recurrent neural network. This non-linear auto regression model with exogenous inputs (NARX model) is fed to a multi-layer perceptron (MLP) which calculates the new output of the RNN. If this results in a large NARX model order, then the network might become too large and a slow down of training occurs. In this case it might be necessary to make the network fully recurrent, which is more powerful in acquiring the system dynamics as the MLP is more powerful in function approximation then a single-layer perceptron.

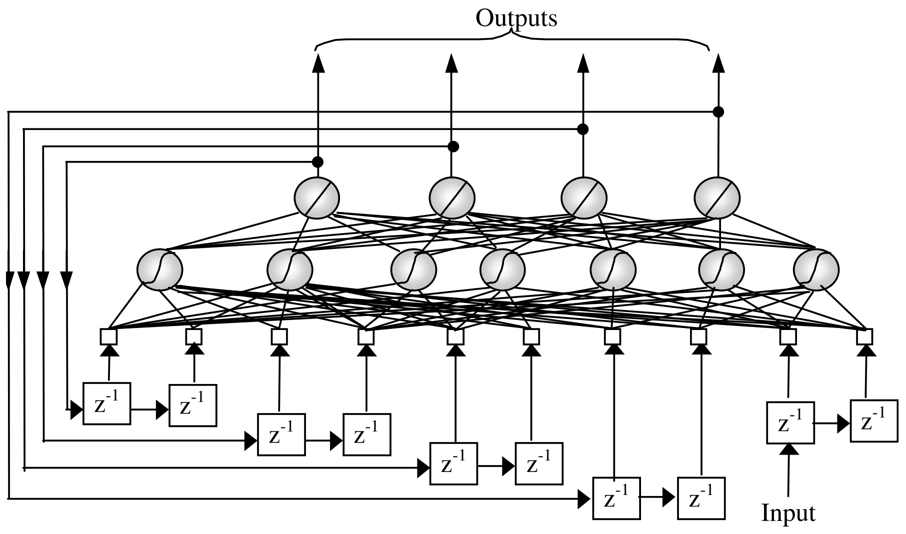


Figure 7: A schematic model of a NARX recurrent neural network of order 1

As for the ANN an error can be declared comparing the output of the RNN with the desired output (d), normally, a quadratic cost function of the error is considered, which can be minimised by the gradient methods mentioned above.

For on-line applications, the optimisation methods usually do not have a recursive calculation scheme and cannot be used, but the backpropagation algorithm is typically slow and forgets the past data. A system identification method such as the Kalman filter could be used to update the network parameters, which has the advantage to take into account the past data when it calculates a new optimal estimate with the new arrived data.

2.2. Neural Networks and the Kalman Filter

One of the first attempts of training neural networks was conducted by Singhal and Wu (1989). Singhal and Wu used a Global Extended Kalman Filter (GEKF) to train feedforward neural networks having an excellent performance on training the network weights more efficiently at the cost of a large increase in storage and computational requirements. Shah et al. (1992) proposed a Multiple Extended Kalman Algorithm (MEKA) to train feedforwarded neural networks on a classification algorithm. With this algorithm a local Kalman filter is designed for every neuron present in the network. They compared their algorithm with the global extended Kalman filter algorithm and concluded that the MEKA algorithm has similar convergence properties but is computationally less expensive. Though the last algorithm is adopted in this work, both will be shown to give a more complete overview of the training of neural networks with the Kalman filter. Another approach adopted by Puskorius and Feldkamp (1994) is due to the observation that some interactions of the weights can be neglected, which decreases the memory storage of the error covariance matrix of the filter. This approach is the so-called Decoupled Extended Kalman Filter (DEKF); with full decoupling there is no interaction of the weights of one neuron with the weights of another neuron and will thus become comparable to the MEKA algorithm.

The Kalman filter identifies a linear stochastic dynamical system. To be able to estimate parameters with the Kalman filter, the parameters has to be written as a dynamical system. The weights for neuron j in layer k can be written as the following dynamical system:

$$\begin{aligned} w_{k,ji}(n+1) &= w_{k,ji}(n) + q_{k,ji}(n) \qquad i = 0\ldots N_{k-1} \\ y_{k,j}(n) &= f\left(\sum_{i=0}^{N_{k-1}} w_{k,ji}(n)\, y_{k-1,i}(n)\right) + r_{k,j}(n) \end{aligned} \qquad (12)$$

where $q_{k,ji}$ and $r_{k,j}$ are stochastic variables with a Normal random Gaussian distribution, N(0, Q) and N(0,R), respectively.

The stochastic process noise, q, would be in fact zero as a parameter has in its definition no stochastic noise compound. But in training neural networks it was pointed out by Puskorius and Feldkamp (1994) that adding process noise stabilises the Kalman filter and also prevents the algorithm to get stuck in poor local minima. A larger process noise also speeds up the training process.

The Kalman filter provides an elegant and simple solution to the problem of estimating the states of a linear stochastic dynamical system. For non-linear problems the Kalman filter is not strictly applicable as the linearity plays an important role in its derivation. The extended Kalman filter tries to overcome this problem by linearising the stochastic dynamical system about its current state estimation, which first seems to be suggested by Kopp & Orford (1963) and Cox (1964). It should be noted that the extended Kalman filter will not be optimal in general. Moreover, due to the linear approximation, it is quite possible that the filter may diverge and therefore care has to be taken in applying this method (Goodwin and Sin, 1984)

Expansion of the non-linear dynamical system of the neural network weight parameters around the estimate of parameter state vector w(n-1), in this case the weights, at time t leads to:

$$\begin{aligned} w_{k,ji}(n+1) &= w_{k,ji}(n) + q_{k,ji}(n) \qquad i = 0\ldots N_{k-1} \\ y_{k,j}(n) &= f\left(\sum_{i=0}^{N_{k-1}} w_{k,ji}(n)\, y_{k-1,i}(n)\right)\Bigg|_{w=\hat{w},q=0} + C(n)^T \left[\underline{w(n)} - \underline{\hat{w}}(n)\right] + r_{k,j}(n) \end{aligned} \qquad (13)$$

Where C(n) is the Jacobian matrix resulting from the Taylor expansion about the state at time n and is recalculated at every sampling instance. W_∞ is the real paramater and C(n) is given by:

$$C(n) = \left.\frac{\partial f\left(w(n), q(n)\right)}{\partial w(n)}\right|_{w=\hat{w}(n),q=0} \qquad (14)$$

As for the linear stochastic system shown in the appendix, a Kalman filter can be set-up to estimate the non-linear system and is (Goodwin and Sin, 1984):

$$\begin{aligned} w_{k,ji}(n+1) &= w_{k,ji}(n) + K_{k,ji}(n) e_{k,j}(n) \qquad i = 0\ldots N_{k-1} \\ K_{k,ji}(n) &= P_{k,j}(n)\, C(n)^T \Big/ \left(C(n)\, P_{k,j}(n)\, C(n)^T + R(n)\right) \\ P_{k,j}(n+1) &= \left(I - K(n)\, C(n)\right) P_{k,j}(n) \left(I - K(n)\, C(n)\right)^T + K(n)\, R(n)\, K(n)^T \end{aligned} \qquad (15)$$

The update equation for the covariance matrix P is written in most textbooks as:

$$P_{k,j}(n+1) = (I - K(n)\,C(n))\,P_{k,j}(n) \qquad (16)$$

which is a simplification after substituting the formula for the Kalman gain. But care has to be taken with this substitution, because the covariance matrix of Eq. 9 is not positive definitive anymore by definition and can lead to numerical instability. This was one of the main reasons for the non popularity of the filter some decadades ago.

The Kalman filter of Eq. 8 is used together with the forward pass of the backpropagation when there is no process noise present. The forward pass calculates the new estimates of the observations y, after which the innovation (= output – desired output) is used to adjust the weights. If there is process noise present ($Q \neq 0$) then the error covariance matrix has to be updated during the forward pass by:

$$P_{k,j}(n+1) = P_{k,j}(n) + Q(n) \qquad (17)$$

The Kalman filter has to be initiated with initial values for the states and the covariance matrixes, while the matrixes Q and R are tuning parameters and can be chosen to obtain a certain convergence behaviour (see paragraph 2.2.1). The initial value for the states or ANN weight parameters are chosen at random from a normal or uniform distribution resulting in weights ranging from –02 to 0.2. The error covariance matrix is set initially to a matrix with large values, like 100, on its diagonal.

The jacobian matrix C(n) is calculated conveniently by the backpropagation method, but differs for the two different Kalman training algorithms.

The MEKA algorithm uses a Kalman filter for every neuron, thus the weights are adjusted by the local error, which is calculated in the same way as in the backpropation algorithm. This also has the advantage that the denumerator of the Kalman filter equation becomes a scalar and no matrix inversion is needed anymore. The matrix C has a dimension of the number of inputs to the neuron

The GEKF algorithm adjusts all the weights of the neural network by *one* extende Kalman filter. Herefore the weights have to be positioned in W x p matrix, where W is the total number of nodes and p is the number of outputs of the network.

2.2.1 The Tuning Parameters of the Kalman filter

The process noise covariance matrix Q and the measurement noise covariance matrix R are normally regarded as the tuning parameters of the Kalman filter. It was already mentioned that the process noise for a parameter would be zero, but accelerates the learning process and helps to avoid local minima. From Eq. 10 it can be seen that the error of the state estimation will start to grow after the sampling point until the new measurement will arrive. The measurement noise determines how much can be trusted on the new measurement and according to this a correction is made by the Kalman filter gain. So a larger value for R will result in a smaller adjustment for the weights. The matrix R cannot have any zeros on its diagonal, otherwise this leads to a division by zero.

It can also be shown that if the the matrixes Q and R are constant, then the Kalman gain will converge to a constant value. This is exactly what is not wanted for an on-line process identification tool. Therefore a way has to be found to keep the filter excited to new data.

Shah et al. (1992) used a similar formulation as is used for the method of recursive minimum least squares. A forgetting factor is introduced in the minimisation criterion, which results in the following equations for the Kalman filter (Shah et al., 1989):

$$w(n+1) = w(n) + K(n)e(n)$$
$$K(n) = \lambda^{-1} \frac{P_{k,j}(n)\, C(n)^T}{\left(\lambda^{-1} C(n)\, P_{k,j}(n)\, C(n)^T + I\right)} \tag{18}$$
$$P_{k,j}(n+1) = \lambda^{-1}\, P_{k,j}(n) - \lambda^{-1}\, K(n) C(n) P_{k,j}(n)$$

A more elegant way is to make the process noise covariance matrix and/or the measurement noise covariance matrix a function of time. Rivals and Personnaz (1998) made the measurement noise covariance matrix a function of the number of epochs, while maintaining the process covariance matrix 0. The function used for R is an exponential function and is:

$$r(n) = (r_0 - r_f)\, exp\{-\alpha i\} + r_f \tag{19}$$

where r_0 was chosen in the order of one (about the order of the inital mean squared error averaged over the number of data), r_f a small value like 10^{-10}, α 0.5 to 1 and i is the number of epochs.

Though their function is a function of the number of epochs, it is not in a form suitable for on-line training. Therefore it is proposed to make the matrixes a function of the error, in which way the training of a neural network can be controlled nicely and a certain convergence behaviour can be obtained depending on the characteristics needed for the problem. For example, if a certain measurement is very inaccurate, then the measurement covariance matrix can be maintained set to a higher value to give a not so large weight adjustment when the prediction error starts to grow.

Therefore both the process noise covariance matrix and the measurement noise covariance matrix can be made a function of the error. The functions are linear or exponential. The process noise covariance matrix will be larger in the beginning and go to zero when the error decreases. The measurement noise covariance matrix can be made larger in the beginning also to make the weight changes less severe, which is normally wanted in the beginning of neural network training as all weights are non-optimal.

$$q(n) = q_0 \left\lfloor exp\left(1.0 * 10^{-2} * SSE\right) - 1.0 \right\rfloor \tag{20}$$

Where SSE is the sum square error of all the outputs and q will become zero when the SSE reaches zero. q_0 was set to 0.1. In this work the SSE was calculated over the whole batchrun. It would be better to make the error a function of the absolute local error, which will adjust the weights more if its error is high while otherwise the weight change is small.

It was already shown by Scheffer and Maciel Filho (2000) that training with the Kalman filter was a potential candidate of a training algorithm for recurrent neural networks, but that the way the algorithm was implemented was computational to heavy due to the memory requirements. To diminish the memory requirements, the weight matrix of the network was re-structured into a vector by subsequent numbering of the weights. The Kalman filter matrixes were defined from this vector, which reduced the required memory reasonably and made algorithm have calculation time a little bit higher than the conjugate gradient method. The algorithm becomes slower when the networks dimension grows.

- figure to summarise the calculation method

3. Results

The data of the penicillin fed-batch process was obtained from the optimal batch run, which was determined by Rodrigues (1996). The batch run was obtained with a sampling interval of 6 minutes, which is quite large and should be enough for when the neural network would be used in an optimisation scheme. Only a training set was made, as the main objective is to create a on-line identification tool. So there is no need for a test set to check the generalisation properties of the network.

Various recurrent neural networks were trained with variable sizes and different architectures. It was noted that some of the states have a quite linear behaviour, while others exhibit a high non-linear response. To account for both linear and non-linear behaviour a specific network architecture was created, which is as follows:

- The activation function of the output layer is a linear function, while the activation functions of the hidden layer are non-linear and were chosen as the tangent hyperbolic function.
- All the inputs of the recurrent neural network, the re-fed outputs, the input and the time-delayed inputs and delayed outputs, are directly connected to the output layer.

The results shown here are from a recurrent neural network with a NARX order of 1 and the specific architecture mentioned above. The recurrent neural networks without the specific architecture (further on specified by RNNlin) were only able to describe the penicillin process with teacher forcing, when the outputs were re-fed it resulted in high values for the error. It should be mentioned that the error of the Kalman filter was small during the training due to the filtering done during the sequential mode. When the final weights were used for network simulation it resulted in high errors.

The RNNlin networks were much better in modelling the penicillin process, because of their specific architecture. The conjugate gradients method only converges to a small error, when the network is pre-trained to a very small error in the teacher forcing mode. Otherwise the conjugate gradient method gets stuck in a local minima with a much higher error.

Only results from the RNNlin consisting of a hidden layer with 15 neurons with the tangent function and a output layer with 4 neurons with a linear activation function. The order of the NARX model was 1, so the present values of input and outputs and 1 past value of the input and outputs are taken into consideration.

In Figure 8 and Figure 9 the training of this RNNlin is shown for three different implemented training algorithms, actually the method of Levenberg-Marquardt was implemented also but showed itself inferior to the method of conjugate gradients. Probably due to the approximation of the Hessian used or because it is more dependent on the initial shoot.

The error for the sequential mode is much lower, because for every presented sample an adjustment is made for every weight while in the batch training mode the an average adjustment is made over the whole training set.

It can be seen that in the teacher forcing mode the training algorithms behave the same (Figure 8), giving a rapid adjustment in the beginning and afterwards a slower fine-tuning of the weights. The Backpropagation algorithm with momentum is slower as learning parameter is varied with a fixed step-size.

From Figure 9 it shows that when the outputs are re-fed the behaviour of the sequential algorithms is totally different. The neural network has become dynamical

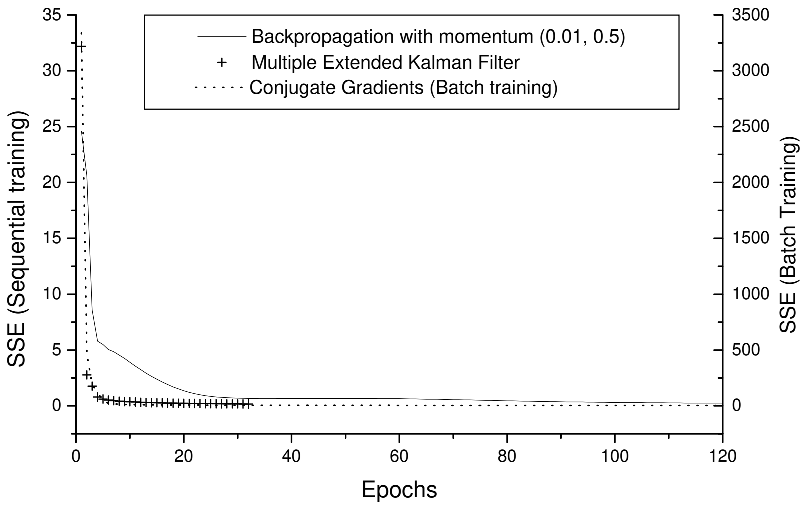


Figure 8: Comparison of the training of a MA1lin-15 network in teacher forcing mode

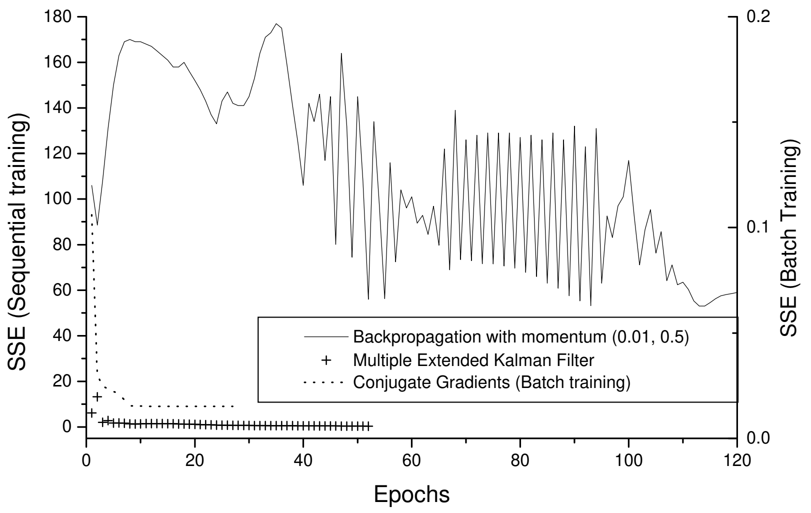


Figure 9: Comparison of the training of a MA1lin-15 network with the network outputs re-fed to the input (network was first trained with teacher forcing after which the outputs were re-fed

which turns the system much more complex for its weight change. The oscillation for the backpropagation algorithm with momentum is due to the momentum term. Though the MEKA algorithm takes more calculation time, it converges very rapidly, making it a very suitable for a on-line training of recurrent neural networks as an system identification tool.

In Figure 10 to Figure 13 the trained RRNlin networks are shown after several apresentations of the training set, one epoch is one apresentation of every training sample of the whole training set. The outputs of the network were re-fed

Very good approximation of the penicillin process is obtained with the RRNlin network, which was trained with the Kalman filter. The experimental data was described perfectly and shows the potential of multiple Kalman filter training algorithm. It should be mentioned that the Linear RNN, which is a network with no hidden layer, the same order of the NARX model and only linear activation functions in the output neurons, describes part of the outputs quite reasonable. The more linear outputs, the biomass concentration and the penicillin concentration, are described accurately, while the non-linear outputs are predicted worst.

The RNNlin trained with the backpropagation with momentum algorithm described the penicillin process the worst. The backpropagation with momentum algorithm was trained to a final summed square error of about 11. It might be that the network can be trained to a smaller error, when the learning parameter is lowered. This augments the training time enormously and therefore was not done as it is not the scope of this work.

The RNNlin trained with the conjugate gradients algorithm is not shown because it coincidences with the prediction of the RRNlin trained with the Kalman filter. The conjugate gradient method is a fast training algorithm also, but has the drawback that a good gradient estimate is needed. So every time that new data arrive and the error augments, the network has to be optimised over a time window. This makes the conjugate gradients algorithm inappropriate for on-line training applications.

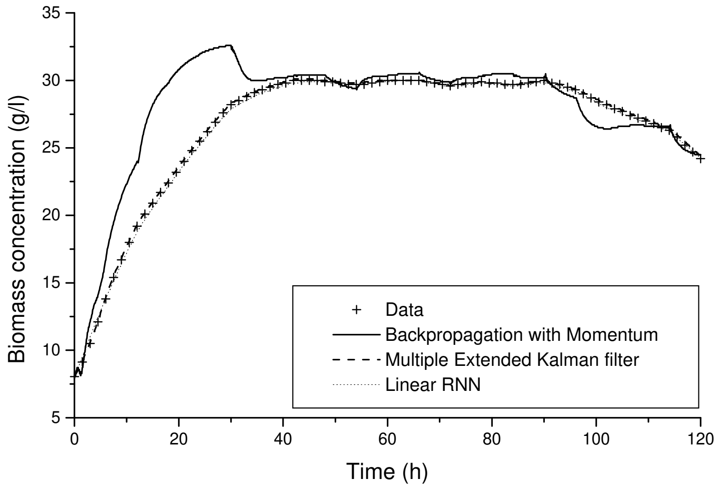


Figure 10: Prediction of the biomass concentration A RNNlin with 15 hidden tanh neurons and 4 output linear neurons trained the different training algorithms with re-alimentation of the outputs after several presentations

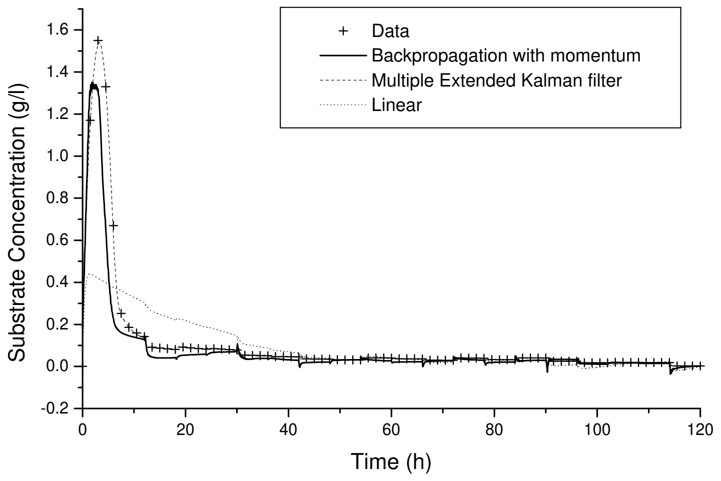


Figure 11: Prediction of the substrate concentration A RNNlin with 15 hidden tanh neurons and 4 output linear neurons trained the different training algorithms with re-alimentation of the outputs after several presentations

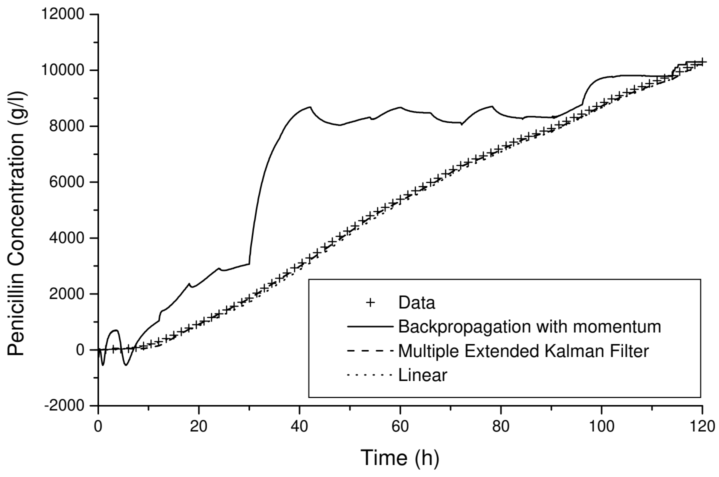


Figure 12: Prediction of the penicillin concentration - A RNNlin with 15 hidden tanh neurons and 4 output linear neurons trained the different training algorithms with re-alimentation of the outputs after several presentations

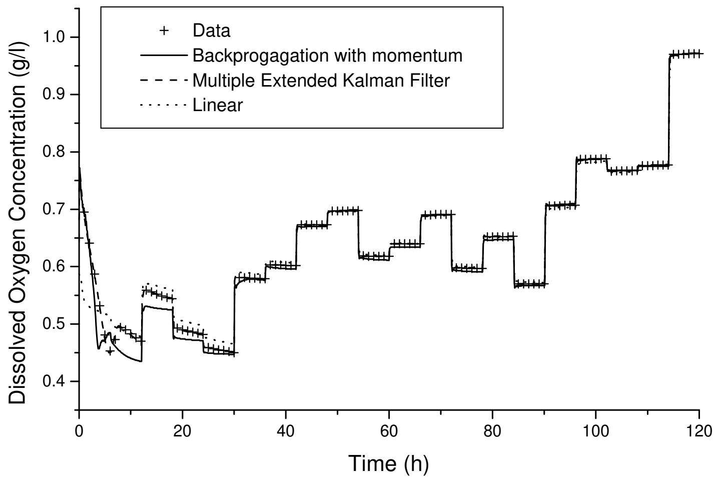


Figure 13: Prediction of the dissolved oxygen concentration - A RNNlin with 15 hidden tanh neurons and 4 output linear neurons trained the different training algorithms with re-alimentation of the outputs after several presentations

Finally a on-line training test was conducted with the RRNlin. Both sequential training algorithms were used to learn the dynamics of the fed-batch penicillin process. The RNNlin was not pre-trained by using teacher-forcing, but directly subjected to training.

In Figure 14 and Figure 15 the on-line training is shown of the RNNlin, predicting the biomass concentration and the dissolved oxygen concentration. The RNNlin weights are adjusted by the filter rapidly and gives a quite reasonable estimation of the penicillin fed-batch process regarding that the RNNlin was never trained before. The peaks are caused by the steps in the dissolved oxygen concentration, which excite the filter to give a rapid adjustment to describe the dissolved oxygen concentration.

It shows that when the Kalman filter is used as a training algorithm no off-line training is necessary if small deviations are allowed for the process to operate apropiately. But in biochemical processes it is vital to have little or no deviation, as a small variation affects the organism and the activity of enzymes.

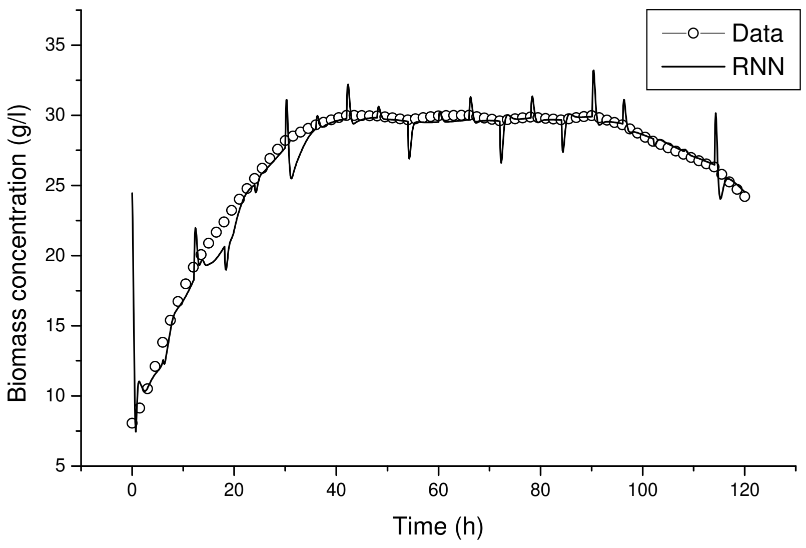


Figure 14: Prediction of the biomass concentration - On-line training of a batch run of the RRNlin with the MEKA algorithm

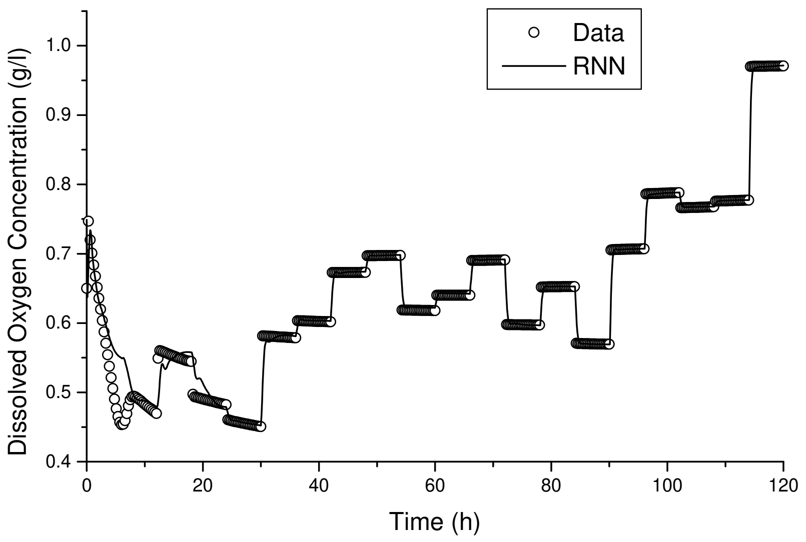


Figure 15: Prediction of the dissolved oxygen concentration - On-line training of the RRNlin with the MEKA algorithm

The on-line training of the RNNlin with the backpropgation algorithm is shown in Figure 16 and Figure 17. The backpropagation algorithm cannot cope with on-line training of recurrent neural networks, as the dynamics of the process are not followed and the prediction is really bad. If the backpropagation is used on-line there has to be trained two networks, one is used to predict the process, while the other is trained off-line to learn the changes. Switching of the networks will keep the networks updated.

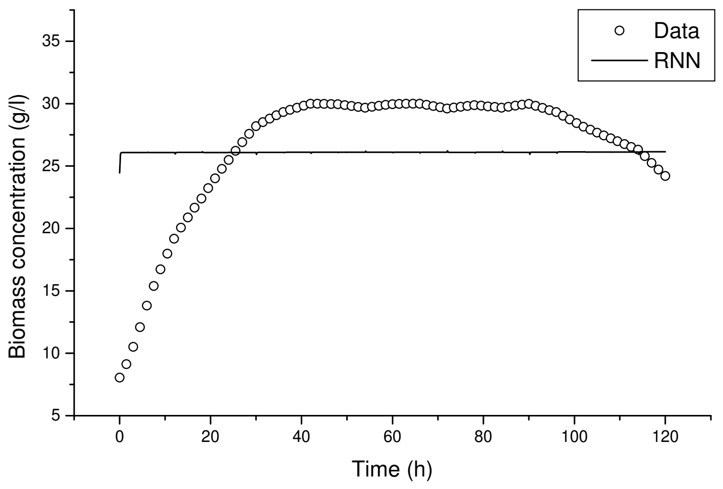


Figure 16: Prediction of the biomass concentration - On-line training of a batch run of a RRNlin with the backpropagation with momentum algorithm

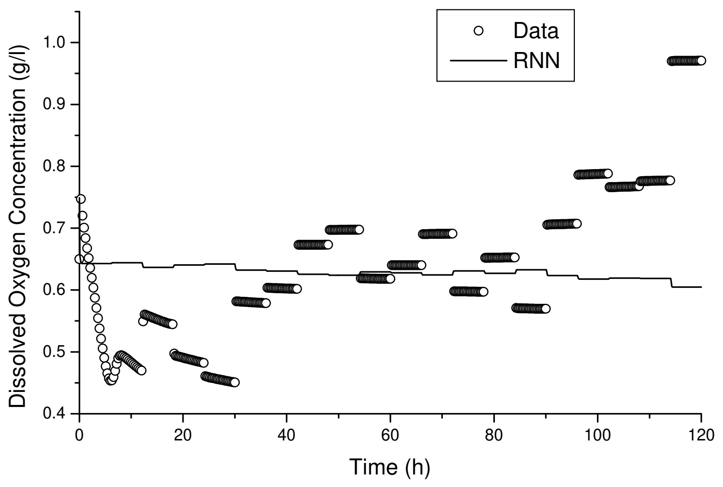


Figure 17: Prediction of the dissolved oxygen concentration - On-line training of a batch run for MA0-lin with the backpropagation with momentum algorithm

4. Conclusions

It was shown that the Multiple Extended Kalman filter is a very powerful training algorithm, especially when the networks are dynamical. Good process descriptions were obtained with a recurrent neural network which has direct connections from the inputs to the outputs. The extended Kalman filter can be used in on-line training schemes, giving reasonable process estimations throughout the process, even when the network was not trained before.

References

Cox, H., *IEEE Trans. Autom. Control*, **AC-9** (1964), 5-12, “On the estimation of state variables and parameters for noisy dynamic systems”.

Crueger, W. and A. Crueger, *Biotechnology: a Textbook of Industrial Microbiology*, (Suderland, Sinawer Associates, Inc., 1984)
Goodwin, G.C. and K.S. Sin, *Adaptive Filtering Prediction and Control* (Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984) 284.
Kopp, R.E. and R.J Orford, *AIAA J.*, **1**(10) (1963), 2300-2306, “Linear regression applied to system identification and adaptive control systems”.
Puskorius, G.V. and L.A. Feldkamp, *IEEE Transactions on Neural Networks*, **5**(2) (1994), 279-297, “Neurocontrol of Nonlinear Dynamical Systems with Kalman filter Trained Recurrent Networks”.
Rivals, I. and L. Personnaz, *Neurocomputing*, **20**(1-3) (1998), 279-294, “A recursive algorithm based on the extended Kalman filter for the training of feedforward neural networks”.
Shah, S., F. Palmieri and M. Datum, *Neural Networks*, **5** (1992), 779-787, “Optimal Filtering Algorithms for Fast Learning in Feedforward Neural Networks”.
Singhal, S. and L. Wu, *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*. (1989) 1187-1190. Glasgow, Scotland: IEEE Press, “Training feed-forward networks with the extended Kalman filter”

Acknowledgements
The authors would like to thank CAPES for the financial support in form of a scolarship.

Appendix

In this appendix it is shown the derivation of the Kalman filter for a one-dimensional linear system. The Kalman filter is a very simple, yet powerful method to estimate states like temperatures and so on from a noisy signal and a convenient way to estimate parameters very easy and rapidly in a recursive fashion.

Suppose a very simple linear stochastic process, which has the simple basic equation of

$$x(n+1) = \phi\, x(n) + v(n) \tag{21}$$

where v is random variable with a normal distribution of (0,Q). In this example ϕ is chosen as 0.9 and Q is 100.

Of the process an error can be defined and is estimated by:

$$P_k(n) = E\left\{(x(n) - \hat{x}(n))^2\right\} \tag{22}$$

Here x(n) would be the real value of x and $\hat{x}$ the estimation of x, which is done from eq. 13. There is no knowledge of the process error, so it is simply initiated with a high value, for example 40000. On time 0, it is measured that the process variable, x_0 is 1000, with an error P. An estimation for x(1) at time 1 can be calculated from 13. As there is no idea of what would be the value of the stochastic variable v, to the best knowledge it can be estimated by its mean of 0.

The real value equation of x(1) is given by:

$$x(1) = \phi\, x(0) + v(0) \tag{23}$$

While the estimated value of x(1) is given by:

$$\hat{x}(1) = \phi\, \hat{x}(0) + \hat{v}(0) = \phi\, \hat{x}(0) \tag{24}$$

The estimation of error for x(1) can be written as a function of the error of x(0):

$$P^-(1) = E\left\{(x(1) - \hat{x}(1))^2\right\} = E\left\{(\phi x(0) + v(0) - \phi\hat{x}(0))^2\right\}$$

$$P^-(1) = E\left\{(\phi(x(0) - \hat{x}(0)))^2\right\} + E\left\{(v(0))^2\right\} + 2\phi\, E\left\{(x(0) - \hat{x}(0))v(0)\right\} \tag{25}$$

The minus is to indicate that this is a calculation before the measurement update. The first term can be identified as the covariance of P(0) the second term as the variance of the process noise Q and the last term is zero as x and v are independent of each other. Eq 17 can be generalised for P(k+1) as a function of P(k) by:

$$P^-(k+1) = \phi\, P^-(k)\phi^T + Q \tag{26}$$

The error for the simple example becomes P(1) = 32000. Now a measurement is done of variable x and is measured with a certain error and suppose that this is given by:

$$y(k) = C\, x(k) + w(k) \tag{27}$$

w is again a random variable and normal distributed by (0,R), here C is 1 and R 10000. Assume that the measurement has a value of 1200. The estimate of x can now be updated by the measurement.

$$\hat{x}^+(k) = \hat{x}^-(k) + K(k)\left\{y(k) - C\,\hat{x}^-(k)\right\} \tag{28}$$

To calculate the optimal value for the Kalman gain K the error is calculated for the state adjusted by the measurement:

$$P^{+}(1) = E\left\{\left(x(1) - \hat{x}^{+}(1)\right)^{2}\right\} = E\left\{\left(x(1) - \hat{x}^{-}(1) - K\left(y(1) - C\hat{x}^{-}(1)\right)\right)^{2}\right\}$$

$$P^{+}(1) = E\left\{\left(x(1) - \hat{x}^{-}(1) - K(1)\left(Cx(1) + w(1) - C\hat{x}^{-}(1)\right)\right)^{2}\right\}$$

$$P^{+}(1) = E\left\{\left((1 - K(1)C)\left(x(1) - \hat{x}^{-}(1)\right) + K(1)w(1)\right)^{2}\right\}$$

$$P^{+}(k) = E\left\{(1 - K(k)C)P^{-}(k)(1 - K(k)C)^{T} + K(k)w(k)K(k)^{T}\right\} \qquad (29)$$

The optimal Kalman gain to adjust the weight can be obtained by differentiation of the error covariance equation to K and setting the derivative to zero, which leads to the known equation for the Kalman gain.

$$K(k) = {C^{T}P(k)} \Big/ {\left(C^{T}P(k)C + R\right)} \qquad (30)$$

The Kalman gain for our simple problem is 0.7647. The adjusted estimation of x = 1129 and the new error P is 7647. From this it can be seen that the error drops very fast and that the Kalman filter is a very efficient estimator.

The process noise covariance matrix of the Kalman filter is important for the parameter adjustment of the Kalman filter, but is normally small. Utilizing a function for the process covariance matrix, did not show an improvement and is therefore intuitively only.

3. O projeto de um controlador não linear da variância mínima – Aplicação no processo semi-batelada de Penicilina

Resumo – Este capitulo mostra um controlador auto-ajustavel de tipo da variância mínima. A identificação não linear é feita por uma rede neural recorrente (RNN), cujos parâmetros foram ajustados por o algoritmo de filtro de Kalman em tempo real. A predição de RNN é alimentado a um filtro de Kalman estendido, que estima os parâmetros de controle ou a variável de controle direta. Este controlador de variância mínima foi testado com o processo semi-batelado de produção de penicilina, de qual se controlou a concentração de oxigênio por manipulação da velocidade de rotação do agitador. A RNN nunca foi treinado antes, mas mesmo assim, o filtro de Kalman ajusta os pesos de RNN eficiente resultando em uma identificação de sistema eficaz. Mesmo sob mudanças severas de processo e ruídos complexos gerados por um ruído fracionário Gaussiano, o controlador auto-ajustavel se mostrou estável e calculando passos de controle adequado, sempre mantendo a concentração de oxigênio perto de seu consigne. Portanto, este controlador auto-ajustável demonstra um potencial para ser aplicado em esquemas avançadas de controle como alternativa de controle preditivo.

Este assunto foi apresentado e publicado nos anais de **6th world congress on chemical engineering em Melbourne, Australia (2001)**

A NEW DESIGN OF A NON-LINEAR SELF-TUNING CONTROLLER - APPLICATION TO A FED-BATCH PENICILLIN PROCESS

R. Scheffer, J.A.D. Rodrigues and R. Maciel Filho

It this paper is presented a non-linear self-tuning controller of the indirect type. A non-linear state identification is performed by a recurrent neural network (RNN), whose parameters are updated by an extended Kalman filter in real-time. The RNN prediction is fed to an extended Kalman filter, which estimates the controller parameter or manipulated control variable. The non-linear self-tuning controller was applied to a fed-batch penicillin process, controlling the dissolved oxygen concentration by manipulating the rotation speed. The RNN was never trained before, but the Kalman filter training algorithm adjusts the RNN weights efficiently, which results in a fast process identification The proposed non-linear self-tuning controller was subjected to severe process changes and a difficult measurement noise generated by a fractional Gaussian noise. The non-linear self-tuning controller showed itself as being robust and able to cope with the process changes, keeping the dissolved oxygen concentration close to its set-point. The self-tuning controller shows a great potential for the use in batch processing and continuous processing, when a high quality control is demanded.

INTRODUCTION

Fed-batch processing is a typical example of a process exhibiting non-linear process dynamics. Especially, biochemical processes are known to have a lot of interaction between their state variables and are sensible to minor changes in pH, dissolved oxygen concentration, and temperature due to the sensitivity of the biochemical catalysts. PID control behaves well in case of continuous processing but in batch processing the control parameters will never maintain optimal values due to the changing process conditions. Therefore a controller implementation needs the self-tuning concept. In most self-tuning concepts the model used is linear, which needs to be updated also. In this work it is proposed a non-linear self-tuning controller, which is based on neural networks. Neural networks are known to approximate any non-linear function and when no new measurement data is available, this will result in a better process estimation than in case of a linear model.

THE CONTROLLER

The proposed controller structure is shown in Figure 18

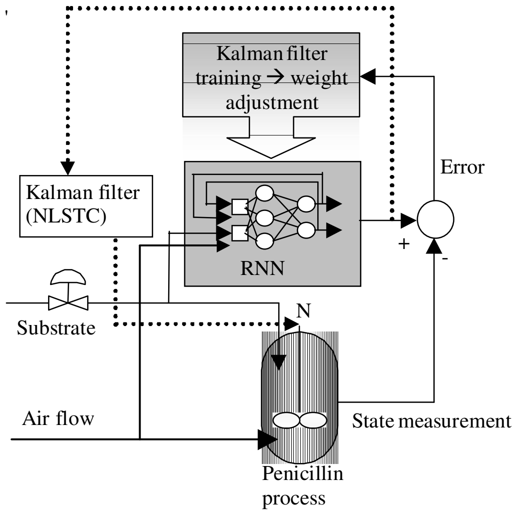


Figure 18: The proposed non-linear Self-tuning controller scheme

The neural network used is a recurrent neural network with direct connections from the input to the output layer, where the activation function in the output layer is a linear function. The linear function accounts for the linear part of the data. The network weights are updated in real-time by a Multiple Extended Kalman filter algorithm to account for changes in the process, whose implementation can be found in Scheffer et al. (2000, 2001b). The only difference between the algorithm described in Scheffer et al. (2001b) and this work is the update of the process noise covariance matrix is which is done here by the local error of every neuron in the network which is described by:

$$Q = q_0(error\ neuron\ i)^2 \tag{3.1}$$

But in this way the measurement noise is included in the process noise, and therefore it might be better to use:

$$Q = q_0(error\ neuron\ i)^2 - var(measurement\ noise) \tag{3.2}$$

If Q is lower than zero it will be set to 0.

In this way, the dependence of the process covariance noise on the error can be seen as a type of covariance resetting as explained by the dynamical update of the Kalman filter equation:

$$P(k+1) = P(k) + Q(k) \tag{3.3}$$

Where the diagonal elements of the error matrix P are augmented by the elements of Q. Care has to be taken to not add to much to the diagonal elements of the error matrix P, because the Kalman filter updates the weights of the recurrent neural network, which would not have a stochastic component. If Q is too high, the noise will be fitted also, reducing the noise filtering capabilities of the Kalman filter.

The estimate of the recurrent neural network is fed to an extended Kalman filter to estimate the controller parameters or directly the manipulated variables. Here, the latter approach is chosen and the manipulated variable is directly estimated by the following dynamical system:

$$m(k+1) = m(k) + w(k) \tag{3.4}$$

$$y_c(k) = y_{ann,c}(k) + v(k) \tag{3.5}$$

where m is the manipulated variable, y_c the controlled variable, w and v are variables with a Gaussian distribution of (0,Q) and (0,R) respectively.

The measurement, d, of the controlled variable is the desired set-point of the controlled variable. The manipulated variable is one of the inputs of the recurrent neural network. In the application of the Kalman filter, the observation equation has to be linearised every sampling instance. Thus the derivative of the controlled variable to the manipulated variable has to be calculated, which is the derivative to the recurrent neural network. The derivative of a neural network can be calculated by applying the chain rule, which results for a two layer neural network in:

$$\frac{dy_{k,j}}{dy_{k-2,h}} = f'\left(v_{k,j}\right) * \sum_{i=1}^{N_{k-1}} \left\{ w_{k,ji} * w_{k-1,ih} f'\left(v_{k-1,i}\right) \right\} \tag{3.6}$$

The controlled variable can now be updated by the Kalman filter:

$$m(k) = m(k) + K(k)\left(y_{c,setpoint}(k) - y_{ann,c}(k)\right) \tag{3.7}$$

THE PENICILLIN PROCESS

The production of antibiotics for clinical applications is an important industrial process and about 60-65% is produced by means of fermentation. The penicillin is produced by bacteria or fungi and in this work the process studied is the production of Penicillin G, which is done by the fungi Penicillium chrysogenum.

The emphasis is put on the production phase and not on the growing phase where an optimal feeding strategy is essential in obtaining a high concentration of penicillin. But it is essential to keep the dissolved oxygen concentration above 30% to ensure life conditions to the fungi. In this work the feeding strategy determined by Rodrigues (1999) is used and the control objective is to maintain the dissolved oxygen concentration at about 55%. The dissolved oxygen concentration is controlled by manipulating the rotation speed through the mentioned non-linear self-tuning controller. A PID controller will be used to compare the performance of the non-linear self-tuning controller.

An essential part of the non-linear self-tuning controller is the recurrent neural network identification. Three input and four state variables were taken to identify the process and are the substrate feed flow, the rotation speed and the air flow as input variables and the bio-mass concentration, the substrate concentration, the penicillin concentration and the dissolved oxygen concentration as state variables. The mentioned Kalman filter algorithm will be compared to the standard backpropagation algorithm.

RESULTS

The control system was applied to a simulation of a fed-batch penicillin process. A detailed model was obtained from Rodrigues et al. (1999) which was validated with industrial data to simulate the penicillin process. The recurrent neural network used in the non-linear self-tuning scheme was never trained before to encounter its behaviour in real-time training. A complicated noise sequence was added to the process variables and consisted of a generated fractional Gaussian noise which has a long term dependence (Mandelbrot and Van Ness, 1968). Both measures were taken to check the robustness of the developed control algorithm. The applied disturbances to the input variables of the penicillin process are shown in Figure 28.

First it is shown some results from the process identification of the penicillin process with and without control of the dissolved oxygen concentration. The recurrent neural network was trained either with the Kalman filter either with the backpropagation algorithm. From Table 2 it can be seen that the parameters of the Kalman filter have a significant influence on the training behaviour. For example the initial value of the error covariance matrix P, p0, is significant for the initial phase of the RNN, as the neural network is never trained before it is essential to put a large value for P to get a fast initial learning and acceptable process estimation in the beginning. The process noise matrix Q has a large influence on the training behaviour, in the beginning it will ensure a faster convergence of the RNN because the matrix P is maintained large, but at a certain point it will prohibit the RNN to converge due to the large changes applied to the weights and thus the prediction will start oscillating along with the noise, which should in fact be filtered out by the RNN. The measurement noise matrix R would be chosen as the variance of the measurement noise to take into account the measurement noise but the internal neurons of the neural network contribute to the noise also. The measurement noise could for example be divided by the number of the neurons which contribute to the noise of the output variable to obtain a better fit of the data.

Table 2 : Total Summed Square Error (SSE) of all the outputs of the recurrent neural network one batch period with and without control of the dissolved oxygen concentration for various initial conditions of the training algorithms parameters

Algorithm	Paramaters	SSE
kalman filter	p0=100, q0=1e-3, r0=1e-6	8.26E+08
kalman filter	p0=100, q0=1e-1, r0=1e-6	3.54E+08
kalman filter	p0=100, q0=1e-1, r0=1e-1	3.54E+08

Algorithm	Paramaters	SSE
kalman filter	p0=1, q0=1e-1, r0=1e-1	1.53E+09
kalman filter	p0=1, q0=1e-1*error^2, r0=1e-1	2.17E+08
kalman filter	p0=100, q0=1e-2, r0=1e-3	8.46E+08
kalman filter	p0=100, q0=1e-2*error^2, r0=1e-3	7.45E+08
kalman filter	p0=100, q0=1e-3*error^2, r0=1e-3	8.30E+08
kalman filter	p0=100, q0=1e-3*error^2/0.05, r0=1e-3	8.26E+08
kalman filter	p0=100, q0=1e-3*error^2/0.05não lin, 1e-3 linear, r0=1e-3	2.17E+08
kalman filter	p0=100, q0=error^2/0.05, r0=1e-2	2.17E+08
kalman filter	p0=100, q0=error^2/0.05, r0=5e-2	2.17E+08
backpropagation	learning parameter = 0.01	6.63E+08
backpropagation	learning parameter = 0.03	2.64E+08
STC/kalman filter	P0=100, q0=0.1*error^2/0.05, r0=5e-2 qrps=error^2	5.52E+08
STC/kalman filter	P0=100, q0=error^2/0.05, r0=5e-2 qrps=0.1*erro^2	4.48E+08
*STC/kalman filter	P0=100, q0=0.01*error^2/0.05, r0=5e-2 qrps=erro^2	8.37E+08
STC/kalman filter	P0=100, q0=(error^2-0.05)/0.05, r0=1e-2 qrps=erro^2	1.09E+09
STC/backpropagation	Learning parameter = 0.001	4.94E+09
*STC/Backpropagation	Learning parameter = 0.004	1.73E+09

The non-linear self-tuning controller for both neural network training algorithms are shown is shown in Figure 19 to Figure 26. The used initial conditions for both training algorithms are marked with an (*) in Table 2. It can be seen that the non-linear self-tuning controller with the Kalman filter controls the system much better than in case of the non-linear self-tuning controller with the backpropagation algorithm. It should be noted that the behaviour of the non-linear self-tuning controller with the Kalman filter depends very much on the values and functions used for the process and measurement covariance matrixes. It appears that too large values for Q are not desirable and thus not result in an approval of the estimation of the RNN along the batch period. It is better to adjust the controller variable more severe. For the chosen settings, it can be seen that the processes becomes better controlled along the batch period and that the RNN estimation of the dissolved oxygen concentration becomes closer to the real concentration. It is also evident that the self-tuning algorithm should not be turned on in the first instances as this leads to a very bad control due to the bad estimation of the dissolved oxygen concentration.

It is evident that the non-linear self-tuning controller gives a much worst control than the Kalman filter, although the state estimation of the non-controlled variables is quite good. The learning parameter had to be chosen about a 10 times smaller because of the more complicated dynamical behaviour. Due to the self-tuning concept there is created one more feedback loop for the recurrent neural network, that is the rotation speed is now a function of the dissolved oxygen concentration. Thus the dissolved oxygen concentration is now a function of two variables turning the estimation problem to difficult for a simple estimation algorithm as the backpropagation algorithm justifying the use of a more computational demanding estimation method as the multiple extended Kalman filter algorithm.

It can be observed that the backpropagation algorithm is not a bad choice for training the neural network either, but probably will need a low pass filter to annihilate the noise characteristics.

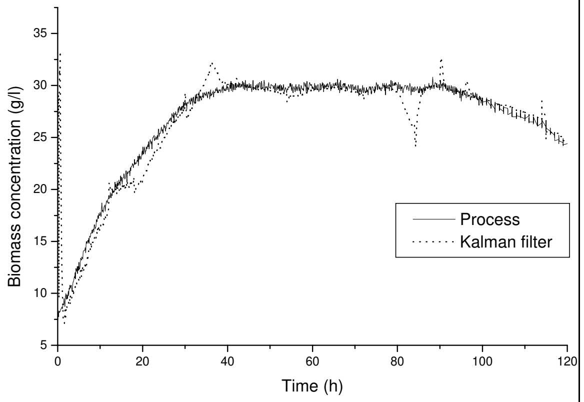


Figure 19 : Estimation of the biomass concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the multiple extended Kalman filter algorithm

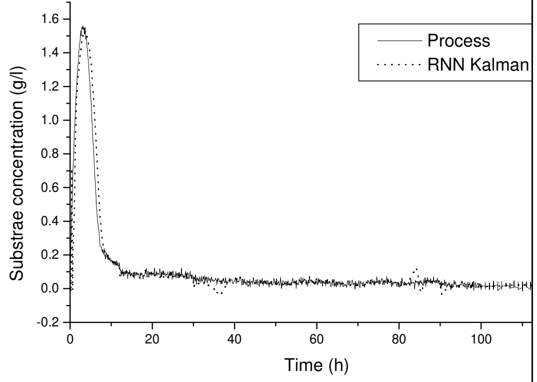


Figure 20 : Estimation of the substrate concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the multiple extended Kalman filter algorithm

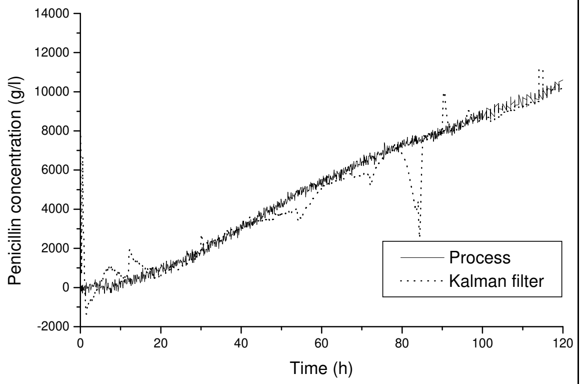


Figure 21 : Estimation of the Penicillin concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the multiple extended Kalman filter algorithm

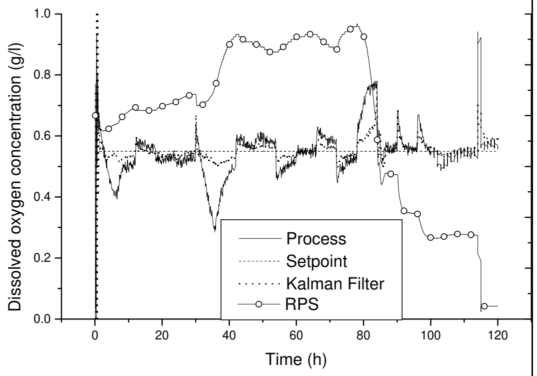


Figure 22 : Estimation and control of the dissolved oxygen concentration concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the multiple extended Kalman filter algorithm

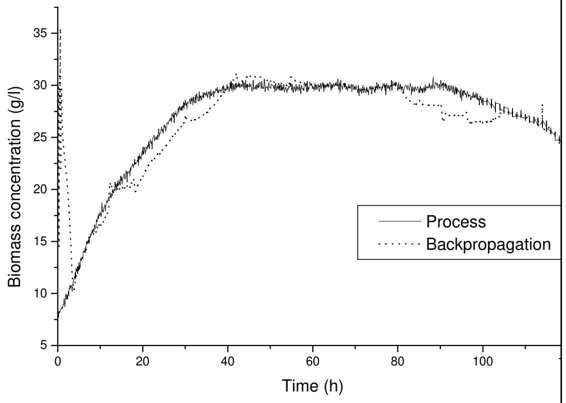


Figure 23 : Estimation of the biomass concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the backpropagation algorithm

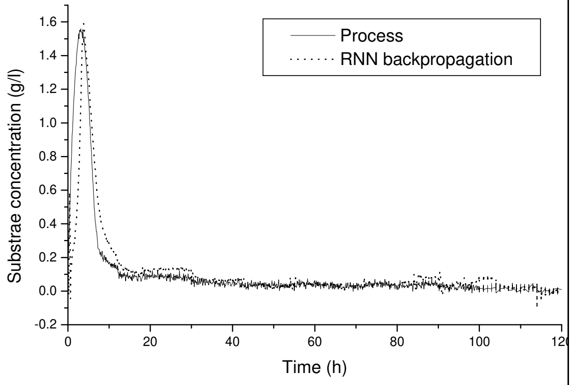


Figure 24 : Estimation of the substrate concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the backpropagation algorithm

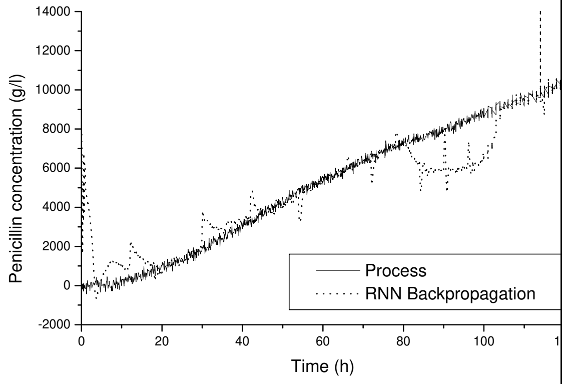


Figure 25 : Estimation of the penicillin concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the backpropagation algorithm

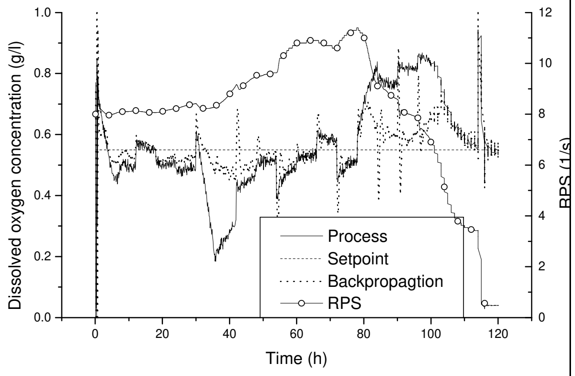
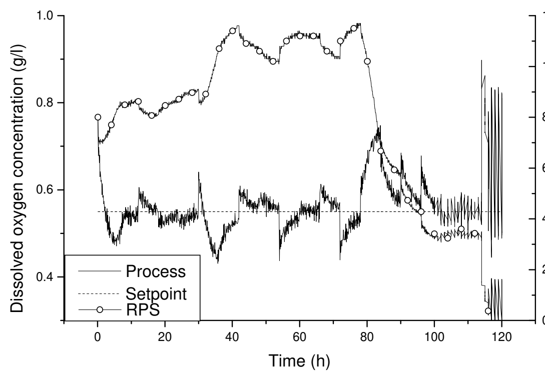
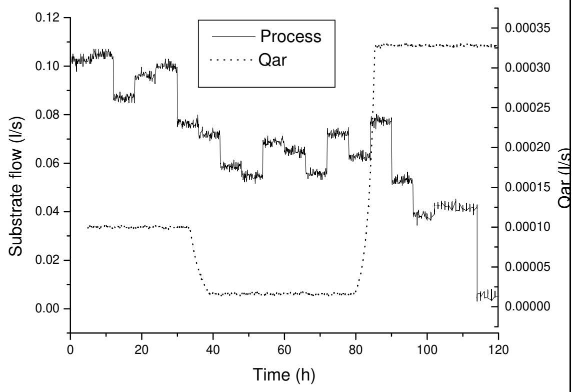


Figure 26 : Estimation and control of the dissolved oxygen concentration concentration of the non-linear self-tuning controller, where the recurrent neural network is trained with the backpropagation algorithm

In Figure 27 it is shown the implementation of a PI controller. It can be seen that if it is wanted to control the dissolved oxygen concentration only a PI control might be satisfactory, because the proportional and integral action were not fully optimised. Observe that it is necessary to filter the dissolved oxygen signal to obtain a smoother control. From Figure 22 it is depicted that the non-linear self-tuning control will lead to a very good control when the recurrent neural network is fully trained. When the recurrent neural network is fully trained, it could be used in a feedforward self-tuning control scheme, where the prediction of the neural network can already annihilate the disturbances.

 Figure 27: PI control of the dissolved oxygen concentration with Kc=5 and τi=1.	 Figure 28: The perturbations applied to the input substrate flow and the airflow.

CONCLUSIONS

It was shown that in non-linear control application, such as controlling a fed-batch process, it is important to develop an advanced control algorithm as a PID controller cannot do the job properly. The developed non-linear self-tuning controller shows an excellent control of the dissolved oxygen concentration of the penicillin production process. The non-linear self-tuning controller showed itself robust in a difficult regulator problem and under noisy measurement conditions. Training of the recurrent neural network, which is the non-linear identification tool of the non-linear self-tuning controller, should be done preferentially with an advanced training algorithm as the extended Kalman filter algorithm instead of the backpropagation algorithm. The changing manipulated variable leads to difficult dynamical system behaviour, but the extended Kalman filter algorithm can successfully cope with it.

REFERENCES

1 Scheffer, R. & R. Maciel Filho (2000), 'Training a Recurrent Neural Network by the Extended Kalman Filter as an Identification Tool', Escape-10 symposium proceedings, pp 223-228

2 Scheffer, R. & R. Maciel Filho (2001), 'process identification of a fed-batch penicillin production process - training with the extended kalman filter' (In: " Application of Neural Network and Other Learning Technologies in Process Engineering ", Ed. I.M.Mujtaba and M.A. Hussain, Imperial College Press

3 Rodrigues, J.A.D and R. Maciel Filho (1999), Chemical Engineering Science, 54(13-14), 2745-2751, "Production optimisation with operating constraints for a fed-batch reactor with DMC predictive control "

4 Mandelbrot, B.B. and J.W. Van Ness, (1968) SIAM Rev., "Fractional Brownian Motions, Fractional Noises and Applications", 10(4), 422-437

4. O ruído Gaussiano fracionário em modelagem de sistemas complexos

Resumo – O ruído Gaussiano fracionário (fGn) foi proposto como preditor de sistemas com um comportamento hidrodinâmico complexo, como colunas de bolhas e unidades de craquemento térmico. O sinal de pressão de um piloto de um reator air-lift foi medida para varias vazões de ar e utilisado para ajustar os modelos fGn e os modelos ARMA. O parâmetro de Hurst foi estimado pelo método de estatístico de R/S e pelo método de máxima verossimilhança, onde o último resultou em uma estimativa melhor do parâmetro em visto das estatísticas do sinal como a função de auto-correlação. Demonstrou-se que o fGn é capaz de descrever os sinais de pressão para vazões de ar baixas, mas falha de descrever a função de auto-correlação para vazões altas de ar. Os modelos ARMA são capaz de descrever a função de auto-correlação para todas as vazões, pelo maior número de parâmetros que podem ser ajustados. Portanto, recomenda-se utilizar o modelo ARMA fracionário para ser desenvolvido como modelo geral da identificação de erros de medida.

Este capitulo foi publidado na revista **CHEMICAL ENGINEERING SCIENCE, 56(2), 2001, pp. 707 – 711**, "The fractional Brownian motion as a model for an industrial airlift reactor" *e* ***nos anais do*** **3rd Europian Congress on Chemical Engineering (ECCE-3)**.

The fractional Brownian motion as a model for an industrial Air-lift Reactor

R. Scheffer and R. Maciel Filho

State University of Campinas (UNICAMP), Faculty of Chemical Engineering, Cidade Universitária Zeferino Vaz, CP 6066, Campinas - SP – Brazil, CEP 13081-970

Abstract

The fractional Brownian motion (fBm) was verified as a model to describe the complex behaviour of a commercial air-lift reactor. A pressure signal recorded was analysed and predictions were made with the fBm model. It can be concluded that the pressure signal air-lift reactor exhibits the features of the fractional Gaussian noise and not of the fBm. Modelling of the pressure signal with the fractional Gaussian noise (fGn) can be used for control purposes or fault diagnosis. The characteristics of the commercial air-lift reactor are encountered in various reactors which shows the great potential of the fGn to describe pressure signals in such reactors. As the variations of the pressure signal can be described by the fGn, the fBm is still a potential candidate to describe variations over the longitude of the reactor.

Keywords: Fractional Brownian motion, Fractional Gaussian noise and air-lift reactor

Introduction

The commercial air-lift reactor studied carries out a slow oxidation reaction. The reactor exhibits a complex interacting behaviour between its operating variables and has a complex flow characteristic.

In modelling of air-lift reactors fractal analysis was already used to determine in which regime the reactor is operating. Camarasaa et al. (2000) identified different flow regimes, heterogeneous or homogeneous, by calculating the parameter of Hurst of a pressure signal. Doing so the right parameter dependence can be chosen in rigorous models to predict temperatures and so on of the air-lift filter

Our aim in this research is to develop a model description of the air-lift reactor, which describes the complex behaviour of the hydrodynamics in more detail but does not result in excessive computational loads for on-line applications. The model could then be used in advanced control algorithms to obtain a slightly better control and therefore a reduction in costs. It should also be noted that the modelling of signals could be used in fault diagnosis analysis.

The chosen model is the fractional Brownian motion (fBm) which can be used to give a total process description as in a black-box fitter or to predict a part of the process, such as the hydrodynamics. In the following paragraphs it is outlined the characteristics of the air lift reactor and the fBm to explain why it is such a good candidate to model complex systems with chaotic behaviour, as for example fluidised beds.

Characteristics of the air-lift reactor

The hydrodynamics of the air lift reactor are highly complex and mostly turbulent. Turbulent behaviour is incorporated in for example computation fluid dynamic modelling as a Markovian component on the velocity. In less detailed models, the turbulence is described by multiple ideally mixed volumes. The turbulence effects variables as the mass transfer and the heat transfer directly.

The air-lift reactor consists of a riser and a down comer. In the bottom of the riser, air is dispersed and absorbed in the liquid phase to be used in the oxidation reaction. At the top, the gas and the liquid phase are separated, where the latter phase is recycled to the riser.

The flow of the recycled liquid and the air flow will determine the flow characteristics in the column, whether the phases are distributed homogeneously whether heterogeneously.

Both systems are two phase systems and if one wants to describe the hydrodynamics in more detail one should not regard turbulence as a random or stochastic component and therefore turbulence should be modelled by model which has interdependence between its passes like the fractional Brownian motion or fractional integrated ARMA models. The fBm could be used for example to predict the temperature or a coefficient which is dependent on the turbulence.

At first the idea was to correlate changes in the bottom of the column to the changes observed at the top of the column, because this would be of most interest in modelling the column for control purposes. Especially, because the bubbles inserted at the bottom will diffuse over the column width. This is modelled by the fBm as its definition results in a signal whose variance grows constantly. To incorporate the walls as a physical limit, the fBm signal could be limited also.

Unfortunately, the simple data acquisition simple used did not allow to observe more than one variable in time at the same time. Therefore we could only analyse the pressure signal at a point in a column and the model could be used for fault detection or process monitoring.

The fractional Brownian motion

The fractional Brownian motion was introduced by Mandelbrot and van Ness (1968), who already mentioned phenomena in hydrology or fluctuations in solids as processes to be estimated by the fBm. Actually, the empirical findings of Hurst were the main reason for developing the fBm as an explanation for this phenomena. Hurst (1951) discovered that for a lot of natural time-series the following relation was valid:

$$\left(R/S\right) \propto \Delta t^{-H} \qquad (1)$$

Where R is the maximum range (= max - min) of the summed differences of the time series and S the standard deviation. H is referred to as the Hurst parameter or the parameter of self-similarity.

Mandelbrot and van Ness (1968) defined the fractional Brownian motion as follows:

$$B(t,\omega)-B(t_0,\omega)=\frac{1}{\Gamma\left(H+\frac{1}{2}\right)}\left\{\int_{-\infty}^{0}\left[(t-s)^{H-\frac{1}{2}}-(-s)^{H-\frac{1}{2}}\right]dB(s,\omega)+\int_{0}^{t}(t-s)^{H-\frac{1}{2}}dB(s,\omega)\right\} \quad (2)$$

Where Γ(x) is the gamma function and $B_H(0,\omega) = b_0$. Mostly this initial condition is 0.

In simulation it is better to use a discrete version of the fractional Brownian motion, where the increment of dB can then be written as $n^{-1/2}\xi_i$. ξ_i is a Gaussian variable with zero mean and unit variance. Then, the discrete fractional Brownian increments are given in the discrete case by (Feder, 1988).

$$B(t)-B(t-1)=\frac{n^{-H}}{\Gamma\left(H+\frac{1}{2}\right)}\left\{\sum_{i=1}^{n}(i)^{H-\frac{1}{2}}\xi_{(1+n(M+t)-i)}+\sum_{i=1}^{n(M-1)}\left((n+i)^{H-\frac{1}{2}}-(i)^{H-\frac{1}{2}}\right)\xi_{(1+n(M-1+t)-i)}\right\} \quad (3)$$

Here n is a small number and gives an approximation to the short-term dependence of $B_H(t)$. These n points are the number of Gaussian variables taken into account between the sampling points. M is the number of sampling points used to calculate the long-term dependence. The second term in (3) will drop quite fast to zero as i goes to minus infinity. Since only M integer time steps are included, it is clear that B_H becomes an independent Gaussian process for t >> M. Note also that the simulation of a fBm is a sliding average over a Gaussian process weighted by a power-law.

The exponent H is the self-similarity parameter and will determine the behaviour of the time-series. If H is 0.5, the increments are independent and form a Gaussian noise process. Its integral then forms the well known Brownian motion. If $0.5 < H < 1$ and the increments increase it is more likely that it will continue to increase. This is called a persistent behaviour and is the one of most interest. For $0 < H < 0.5$, it is more likely that the increments will decrease after a period of increase, which is called anti-persistent.

The H of the time-series has to be estimated from experimental data before the fractional Brownian motion can be used as a prediction model for the pressure. The R/S statistic of Hurst is one of the oldest techniques however also one of the easiest to be implemented. The parameter H can be estimated more accurately by the spectral density of the time-series using Fourier analysis or wavelets. As it is a preliminary study, only the R/S statistic was implemented.

R/S statistic

The R/S statistic was introduced by Hurst (1951) when studying data of reservoirs.

The average of a time series for a time period τ is given by:

$$\langle \xi \rangle_\tau = \frac{1}{\tau} \sum_{t=1}^{\tau} \xi(t) \tag{4}$$

The accumulated departure of the time-series from the mean of the time series is:

$$X(t,\tau) = \sum_{i=1}^{\tau} \left\{ \xi(i) - \langle \xi \rangle_\tau \right\} \tag{5}$$

R now is defined as the difference between the maximum and minimum amounts accumulated for the time-series. Thus R is defined as

$$R(\tau) = \max_{1 \le t \le \tau} X(t,\tau) - \min_{1 \le t \le \tau} X(t,\tau) \tag{6}$$

The range R will typically depend on the time span and normally increases when a larger time span is considered.

The standard deviation is estimated from the time-series by:

$$S = \left(\frac{1}{\tau} \sum_{i=1}^{\tau} \left\{ \xi(i) - \langle \xi \rangle_\tau \right\}^2 \right)^{\frac{1}{2}} \tag{7}$$

The rescaled adjusted range R/S is then correlated by equation 1. The parameter H is estimated by standard regression techniques. For most natural phenomena H is larger than 0.5 and close to 0.73.

Results

A pressure signal from an air lift reactor was recorded at different air flow feed rates. The signal was analysed with the R/S statistic to determine its self-similarity parameter. Standard statistic analysis was done, such as the sample auto-correlation. Simulation of the fractional Brownian motion was carried out and compared to the measured pressure signal.

In Figure 1 the fractal or R/S analysis is shown of one of the pressure signals. The points of the lower flow rates lie well on a straight line with a little variation. The calculated parameters of Hurst are shown in Table 1 and is for low flow rates correlated positively with a parameter H of 0.72. For higher air flow rates the parameter becomes closer to 0.5 and thus closer to an independent process. Due to the higher air flow rates the system in the column will become more turbulent.

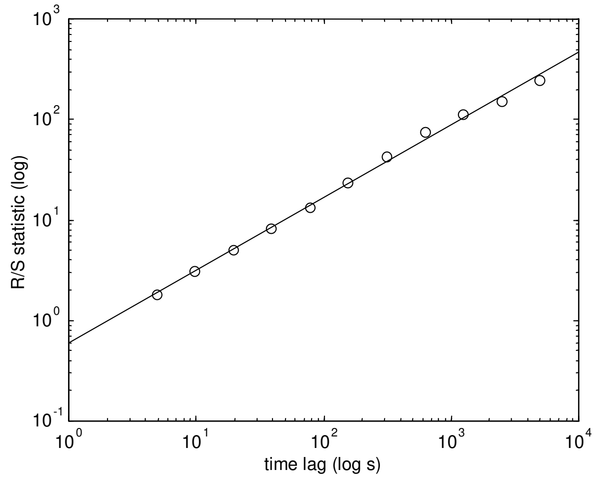


figure 29: Analysis of the pressure signal for na air flow of 3000 with the R/ S statistic

table 3: Estimated Hurst parameter for different air flows

Air flow (n l/h)	Hurst parameter
1000	0.6314
2000	0.7238
3000	0.7218
4000	0.6974
5000	0.6767
6000	0.6623
7000	0.6496
8000	0.6513
9000	0.6329
10000	0.6507
11000	0.6237

	12000	0.5962
	13000	0.5880
	14000	0.5905
	16000	0.5632
	18000	0.5285
	20000	0.5458

From Table 1 it can be noted that the parameter of Hurst is at a maximum for the lower air flow rates and start to drop when the flow rate increases, which is due to the change of the flow regime from homogenous to heterogeneous.

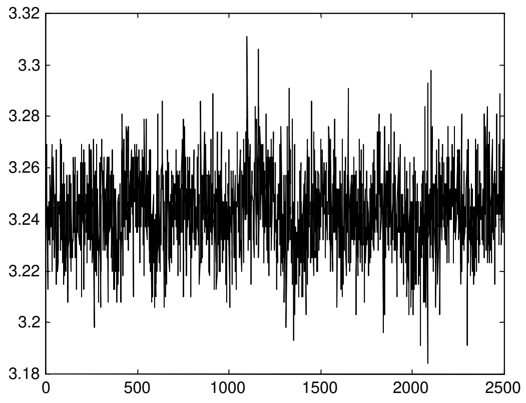


Figure 30: The first 2500 samples of the pressure signal of air-lift reactor for an air flow of 3000

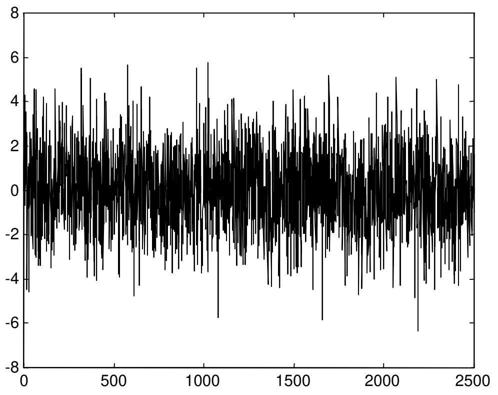


Figure 31: A generated sequence of fractional Gaussian noise with a H = 0.7

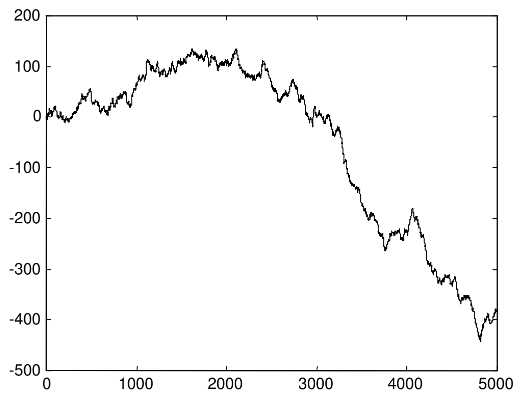


Figure 32: Fractional Brownian motion generated from the fractional Gaussian noirse for H = 0.5

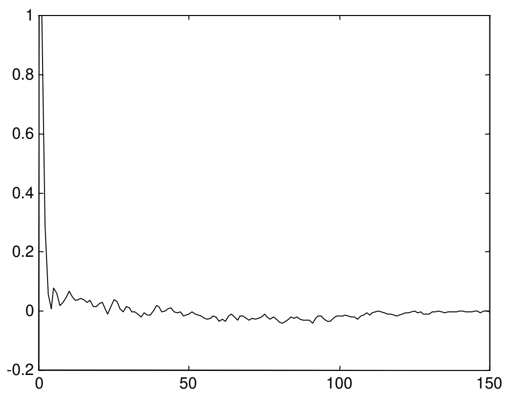


Figure 33 : Sample auto-correlation of the pressure signal for an airflow of 3000 l/h

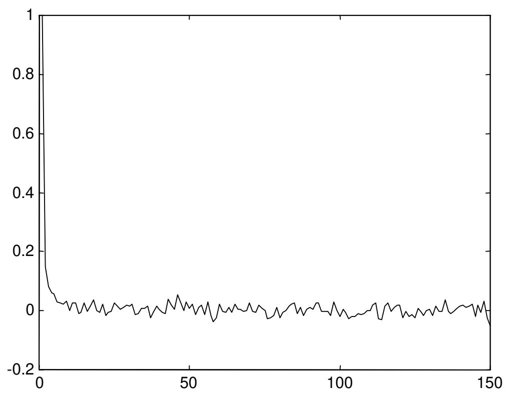


Figure 34: Sample auto-correlation of a simulated fractional Gaussian noise with H = 0.7

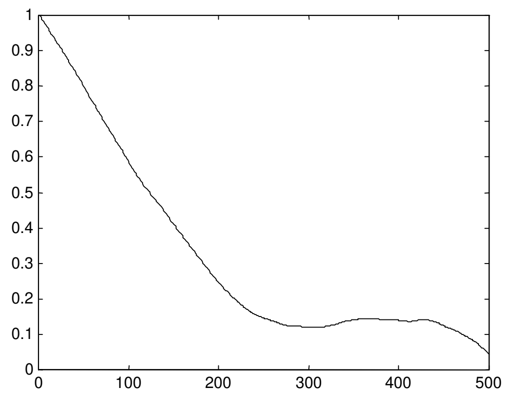


Figure 35: Sample auto-correlation of a simulated fractional Brownian motion with H = 0.7

When looked at the pressure signal and the simulated fractional Brownian motion (Figure 2 and 4), it is very clear that the fBm is not a representative model for predicting the evolution of the pressure signal in time. The pressure signal is a stationary signal, while the fractional Brownian motion is a non-stacionary signal with increasing variance.

By comparing the sample auto-correlation functions of the process with the one of the fBm (Figures 5 and 7) it can be concluded that the pressure signal has a much shorter time memory than the fBm model and cannot be used to described the pressure signal from the air lift reactor.

If the increments of the fBm are regarded, which is called the fractional Gaussian noise (Figure 3) the resemblance is remarkable. The scaling of the fBm was different, because the events were taken from a normal distribution with zero mean and unit variance, but can be easily transformed to obtain the same scaling. It can be seen that the fractional Gaussian noise shows the same kind of variation and the little peaks every now and then.

The sample auto-correlation of the fractional Gaussian noise (Figure 6) shows the same development as the sample auto-correlation of the pressure signal (Figure 5). Both have a short memory, which can be seen from the fast fall for increasing time lags. For large time lags both process and fractional Gaussian noise oscillate around zero, but this cannot be given to much value due to noise effects.

In figure 8 the fractional Gaussian noise is used to estimate a one-step ahead prediction of the pressure signal. Equation 3 was used to predict the pressure signal in time. The increment from t to t+1 was calculated by evaluating the two summations in equation 3. The first summation was calculated by generating ten Gaussian variables with zero mean and a variance of the pressure signal. The second summation in equation 3 was simply done over the time-series of the pressure signal P. This because the increments in the pressure signal are already a result of evaluating the integral at past times. The calculated increment was added to the past pressure data/

It appears that the pressure signal is predicted reasonable with a mean squared error of 0.0219 and mean absolute error of 0.0174. It should be emphasised that the prediction is based on a stochastic process and therefore never will be able to predict exactly the process. Thus it can be concluded that the fractional Gaussian noise is a very suitable model to estimate the pressure signal evolution in time and can thus be used in fault diagnosis programs for the reactor or for control purposes.

The prediction of the pressure signal can be improved by implementing better prediction methods, such as a maximum likelihood estimation, which will be conducted in future work

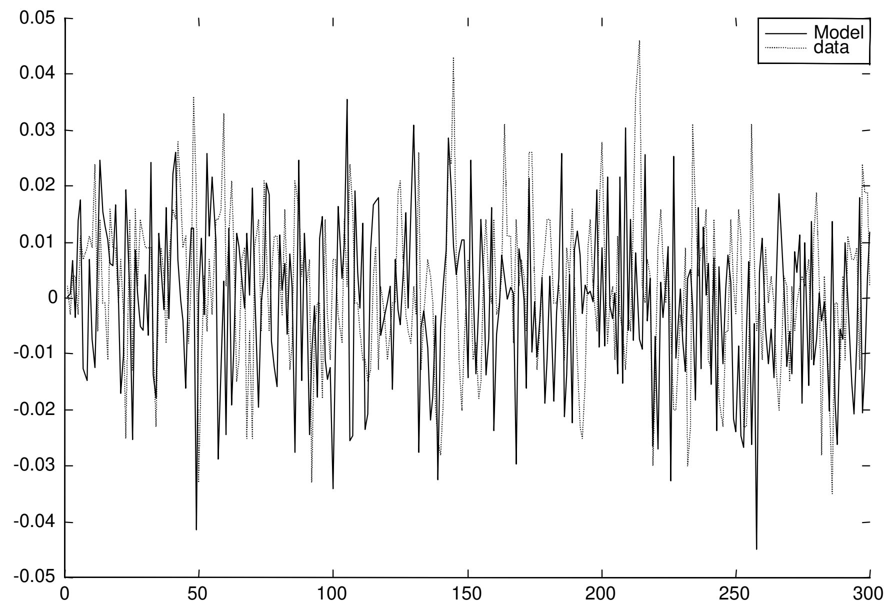


Figure 36: Pressure signal (...) and prediction by the fractional Gaussian noise (-)with H = 0.7218

But the fractional Brownian motion is not out of focus yet as it could be still a very good candidate to model changes taking place along the reactor, predicting for example the changes in time and place in pressure or temperature. This is true as it is shown in this work that the variations measured at a certain height of the reactor are correlated by the fractional Gaussian noise, which is step-size of subsequent points of the fractional Brownian motion.

Conclusions

It was shown that not the fractional Brownian motion but the fractional Gaussian noise is a suitable model to describe the pressure signal evolution in time at a certain height of the air-lift reactor. The fractional Gaussian noise has the same statistics as the pressure signal and can be used as fault diagnosis tool or in control applications.

As the variations of the pressure signal can be described by the fractional Gaussian noise, it is pointed out that the fractional Brownian motion is still a potential candidate to describe variations along the height of the air-lift reactor.

References

Camarasa, E., E. Carvalho, L.A.C. Meleiro, R. Maciel Filho, A. Domingues, G. Wild, S. Poncin, N. Midoux and J. Bouillard, *Development of a complete model for an air-lift reactor, ISCRE-16, submitted (2000)*

Feder, J. *Fractals*, Plenum Press, New York (1988)

Hurst, H.E., *Long-term storage capacity of reservoirs*, Trans. Am. Soc. Civ. Eng., 116, 770-808 (1951)

Mandelbrot, B.B. e J.W. van Ness, *Fractional Brownian Motions, Fractional Noises and Applications*, SIAM Review, Vol. 10, No. 4, pp. 422-437 (1968)

Fractional Gaussian Noise in Modelling Complex Systems

R. Scheffer, R. Maciel Filho
State University of Campinas (UNICAMP, Faculty of Chemical Engineering)
LOPCA/DPQ, Cidade Universitária "Zeferino Vaz" CP 6066, Campinas - SP - Brazil, CP 13081-970

Keywords: Fractional Gaussian Noise, Maximum likelihood estimation, Chaotic systems, air-lift reactor

Abstract

In this work the fractional Gaussian noise (fGn) is proposed as a system identification tool for systems with a complex hydrodynamic behaviour, such as bubble columns and fluid catalytic cracking units. A pressure signal of an industrial air-lift reactor was recorded for various air flow rates and utilised to fit the fGn and ARMA models. The Hurst parameter of the fGn was estimated by the R/S-statistic and by a maximum likelihood method, where the latter method resulted in a better estimate for the parameter when looked at the signal statistics. It is shown that the fGn is a model for the pressure signal at low air flow rates, but fails to describe the auto-correlation at higher air flow rates. The fitted ARMA models are able to describe the auto-correlation for all air flows, but can become non-causal. Therefore these results point out to the Fractional ARMA model would be the preferred model for control purposes as a fault diagnosis tool.

Introduction

Reactors with different phases normally show difficult hydro-dynamical behaviour due to interactions of the phases. The interaction between solids-gas in a fluid catalytic cracking unit or between liquid-gas in a bubble column or in an air-lift reactor results in fluctuations in the pressure signals. The fluctuations of the pressure signal are a result from bubbles passing by the pressure transducers probes and the amplitude of the signal is a function of the bubble size. The size of the bubbles is a complex function of coalescence, bubble break-up, differences in fluid velocities and height in the reactor. Therefore the past of the bubble will typically have an influence on its size.

Fan et al. (1990) applied a fractal analysis, the rescaled range (R/S) analysis, to the pressure fluctuations of a pressure signal of three phase fluidised bed to identify the quality of the fluidization. They show that the Hurst parameter H decreases with increasing gas and liquid flow and increases with height in the reactor. They concluded that the Hurst parameter and fractional Brownian model are proper to analyse the fluidisation of the three phase fluidised bed.

Franca et al. (1991) used the same rescaled range (R/S) analysis to try to identify different flow regimes in two-phase flow in a tube. The Hurst parameter shows that the system is chaotic and the possibility to discriminate plug-flow regime from other flow regimes like annular and slugging flow.

Drahos et al. (1992) studied pressure fluctuations in bubble columns and were able to identify the bubbling regime, homogenous or turbulent bubbling regime, by the rescaled range (R/S) analysis. The Hurst parameter becomes closer to 0.5 for the turbulent regime, indicating that the turbulent region becomes more independent.

Camarasaa et al. (2001) used fractal analysis to determine the flow regime of a pilot air-lift reactor, determining the Hurst coefficient, which indicates whether the bubbling regime of the air lift reactor is in the homogeneous or turbulent regime.

But with the knowledge of the Hurst coefficient the fractional Gaussian noise could be used to predict the fluctuations of the signal, which could be used in advanced model control or as a fault diagnosis tool. The last becoming more important as chemical processes are becoming more and more computerised.

The fGn was originally developed by Mandelbrot and Van Ness (1968) to explain the Hurst phenomenon. fGn is a long-memory model and the correlation between the samples is determined by a self-similarity parameter, H, which is sometimes referred to as the parameter of Hurst.

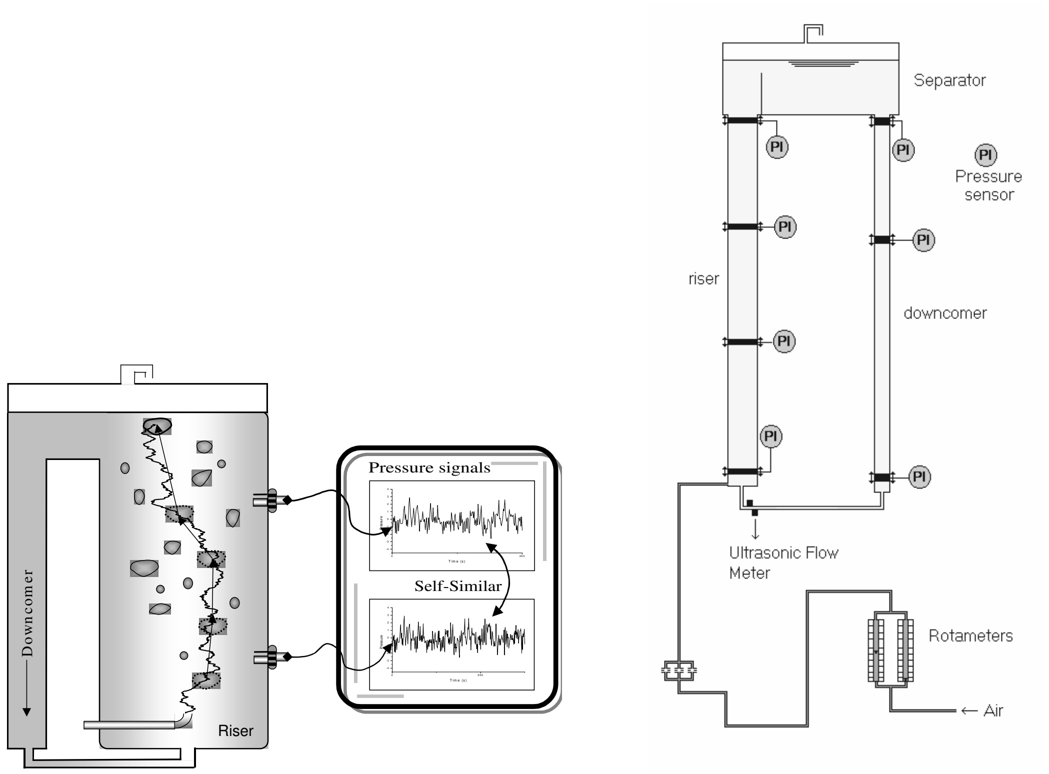


Figure 37: Representation of the fractional Brownian motion and the fractional Gaussian noise applied to an air-lift reactor and a more detailed drawing of the pilot plant reactor used

In Figure 37 the application of the FGN and the fractional Brownian motion (fBm) to the air-lift reactor is shown. The bubbles inserted at the bottom will diffuse over the column width, which is modelled by the fBm as its definition results in a signal which variance grows constantly. To incorporate the walls as a physical limit, the fBm signal could be limited also. The variation of a signal at a certain height is the derivative of the fBm and is called the fractional Gaussian noise (fGn).

Fractional Gaussian noise

Hurst (1951) discovered that for a lot of natural time-series, such as the Nile river and tree rings, the following relation is valid:

$$\left(R/S\right)/\gamma \propto \Delta t^{H} \quad or \quad \left(R/S\right)/\gamma \propto N^{H} \tag{4.1}$$

Where R is the maximum range (= max - min) of the summed differences or cumulative departure of the time series and S the standard deviation. H is referred to as the Hurst parameter or the parameter of self-similarity.

Mandelbrot and van Ness (1968) considered a continuous time process $B_H(t)$, which satisfies the self-similarity property such that for all τ and $\varepsilon>0$, $B_H(t+\tau)-B_H(t)$ has exactly the same distribution as $[B_H(t+\tau\varepsilon)-B_H(t)]/\varepsilon^H$. From this it is clear that the sequential range of B_H will increase in time proportionally to N^H, where the sequential range is defined by:

$$seq.range = \max_{t<r<t+N} B_H(r) - \min_{t<r<t+N} B_H(r) \tag{4.2}$$

where t is the continuous time and H is the model parameter. If the variable $B_H(t)$ is Gaussian, then it is called the fractional Brownian motion. The discrete time fractional Gaussian noise (FGN) is obtained for a discrete time t with a fixed step size:

$$z_t = B_H(t+1) - B_H(t) \tag{4.3}$$

The exponent H is the self-similarity parameter and will determine the behaviour of the time-series. If H is 0.5, the increments are independent and form a Gaussian noise process. Its integral then forms the well known Brownian motion. If $0.5 < H < 1$ and the increments increase it is more likely that it will continue to increase. This is called a persistent behaviour and is the one of most interest. For $0 < H < 0.5$, it is more likely that the increments will decrease after a period of increase, which is called anti-persistent. In modelling a value of H of 0.5 would probably be identified as a highly turbulent system, because of the column being more ideally mixed.

The theoretical Auto-Correlation Function at lag k of the FGN is given by:

$$\rho_k = \tfrac{1}{2}\left[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}\right] \quad 0 < H < 1 \text{ and } k \geq 1 \tag{4.4}$$

From this the N×N correlation matrix of the FGN is given by:

$$C_N(H) = \lfloor \rho_{|i-j|} \rfloor \tag{4.5}$$

To define these matrixes, which can be used in a maximum likelihood prediction, the self-similarity parameter or the Hurst parameter has to be estimated.

The H of the time-series has to be estimated from experimental data before the fractional Gaussian noise can be used as a prediction model for the pressure. The R/S statistic of Hurst is one of the oldest techniques however also one of the easiest to be implemented. The parameter H can be estimated more accurately by the spectral density of the time-series using Fourier analysis or wavelets.

A more sound statistical estimator of H can be determined via the maximum likelihood estimation of Hipel and Mcleod (1994) and is described further on.

R/S statistic

The R/S statistic was introduced by Hurst (1951) when studying data of reservoirs.

The average of a time series for a time period τ is given by:

$$\langle \xi \rangle_\tau = \frac{1}{\tau} \sum_{t=1}^{\tau} \xi(t) \tag{4.6}$$

The accumulated departure of the time-series from the mean of the time series is:

$$X(t,\tau) = \sum_{i=1}^{t} \left\{ \xi(i) - \langle \xi \rangle_\tau \right\} \tag{4.7}$$

R now is defined as the difference between the maximum and minimum amounts accumulated for the time-series. Thus R is defined as

$$R(\tau) = \max_{1 \le t \le \tau} X(t,\tau) - \min_{1 \le t \le \tau} X(t,\tau) \tag{4.8}$$

The range R will typically depend on the time span and normally increases when a larger time span is considered.

The standard deviation is estimated from the time-series by:

$$S = \left(\frac{1}{\tau} \sum_{i=1}^{\tau} \left\{ \xi(i) - \langle \xi \rangle_{\tau} \right\}^{2} \right)^{\frac{1}{2}} \tag{4.9}$$

The rescaled adjusted range R/S is then correlated by Eq. (4.1). The parameter H is estimated by standard regression techniques by plotting the data on a log-log scale. For most natural phenomena H is larger than 0.5 and close to 0.73.

Maximum Likelihood Estimation of the Hurst parameter

The only link between the fractional Gaussian noise and the time-series are formed by the mean, variance and the Hurst parameter, H. By most researchers the R/S-statistic is employed or one of its variants, such as the variance plot. But these methods do not take into consideration any statistical properties of the fractional Gaussian noise. A least squares estimate for H can be formulated by using the sample auto-correlation function, but its efficiency and the theoretical foundations were questioned by McLeod and Hipel (1978).

A more efficient parameter estimation can be developed by employing the Maximum Likelihood Estimation (MLE) method. The following MLE was developed by McLeod and Hipel (1978).

Give a historical time series $z_1, z_2, \ldots, z_n$ the log likelihood of μ, γ_0, and H in the FGN model is:

$$Log\ L_{max}(H) = -\tfrac{1}{2} log\left|C_N(H)\right| - {}^{N}\!\!/_{2}\, log\left[S(\mu, H)\right] \tag{4.10}$$

where $C_N(H)$ is the correlation matrix as defined in Eq. (4.10). In the log likelihood function the norm has to be taken of the correlation matrix and was chosen as the Euclidian norm. The function $S(\mu,H)$ is given by:

$$S(\mu, H) = (z - \mu 1)^T \left[C_N(H)\right]^{-1} (z - \mu 1) \tag{4.11}$$

where z^T equals $z_1, \ldots, z_N$ and is a $1 \times N$ vector and 1^T equals $1, \ldots, 1$ and is a $1 \times N$ vector.

For a fixed H the MLE of the mean μ and variance γ_0 are respectively,

$$\hat{\mu} = \left\{z^T \left[C_N(H)\right]^{-1} 1\right\} \Big/ \left\{1^T \left[C_N(H)\right]^{-1} 1\right\} \tag{4.12}$$

and

$$\gamma_0 = N^{-1}\, S(\hat{\mu}, H) \tag{4.13}$$

The MLE estimate of H, $\hat{H}$, can be estimated from above equations by applying an inverse quadratic interpolation search method to maximise log L_{max}(H). The variance of $\hat{H}$ can be approximated by differentiation twice the log L_{max} by a numerical finite difference method,

$$Var(\hat{H}) = -1 \Big/ \left.\frac{\partial^2 \log L_{max}(H)}{\partial H^2}\right|_{H=\hat{H}} \tag{4.14}$$

As a final check a model adequacy test should be used to see if the model fits the data, which can be done by checking the whiteness of the residuals.

Forecasting with FGN

For most models a mean square error forecast can be defined and thus so for FGN models. For this the covariance matrix, Γ_N, is necessary, which can be obtained by substituting the MLE for H from the above section and dividing by the estimated variance. Then the one step ahead forecast is given by Hipel and McLeod (1994):

$$E\{Z_{N+1} | Z_N\} = \mu + \gamma_N^T \Gamma_N^{-1}(Z_N - \mu 1) \tag{4.15}$$

which can be calculated economically by applying a Cholesky decomposition.

Results

A pressure signal of a pilot plant air-lift reactor was sampled every 0.02 seconds and 10000 samples were recorded per air-flow. The Hurst parameter was estimated for both techniques, the R/S-statistic and the MLE method, where the latter method is done for a number of points only.

In Table 4 the estimated Hurst parameters are shown for both methods and it can be seen that there is a large discrepancy between both methods. It appears that the MLE estimation method gives a different value and *trend* of the Hurst parameter: the MLE estimation results in a low value of H for the low air flow rates, while the R/S statistic gives a high value for H. For high air flows the MLE estimates a high value for H while the RS-statistic gives a low value. The difference in the estimated value of H is due to the statistics of the underlying model. While the R/S-statistic assumes actually no model, the MLE method fits the time-series to the theoretical auto-correlation of the fractional Gaussian noise. As was mentioned by Hipel and McLeod (1994), Hurst should have focussed his research rather on the Rescaled Adjusted Range than on the Hurst coefficient, as one is only a re-scaled function of the other.

Table 4: Comparison of the estimated Hurst parameters by the R/ S-statistic and the Maximum Likelihood Estimation method for the pressure signal of a bubble column

Airflow (n l/h)	H (RS-stat.)	H (MLE of Hipel and McLeod)			
		n=100	n=300	N=500	N=1000
1000	0.6174	0.5587			
2000	0.7199	0.5453			
3000	0.7195	0.5959			0.6001
4000	0.6790	0.6039			
5000	0.6682	0.6143			
6000	0.6659	0.6271			
7000	0.6468	0.6309			
8000	0.6136	0.6474			
9000	0.6223	0.6537			
10000	0.6131	0.6613			
11000	0.5826	0.6687			
12000	0.5778	0.6809			
13000	0.5610	0.6906			
14000	0.5669	0.6930			
16000	0.5498	0.7055	0.7080		
18000	0.4899	0.7111	0.7136		
20000	0.4999	0.7180	0.7228	0.7232	

In Figure 38 the auto-correlation of the pressure signal for an air flow of 3000 nl/h is compared to the theoretical auto-correlations of the fGn. It looks like that the auto-correlations at lower air flows (around 3000 nl/h) of the pressure signal (Figure 37) exhibit the dependence of a theoretical auto-correlation of the fGn and it appears to fall between the theoretical auto-correlations of fGn with H=0.5 and H=0.6. Thus the estimation of the Hurst coefficient by the R/S-statistic is rather high, compared to the Hurst coefficient indicated by the auto-correlation. In (Figure 39) it is shown why the R/S-statistic differs so much from the MLE method and is for short ranges comparable to the MLE estimate.

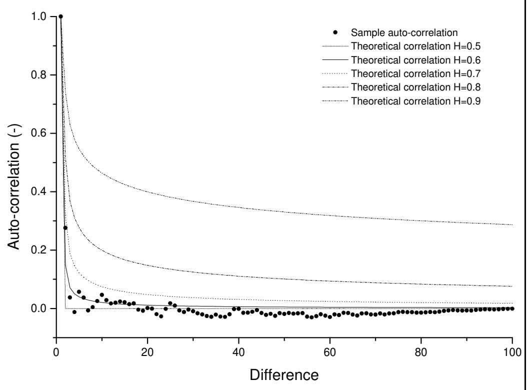


Figure 38: Auto-correlation of the pressure signal with an airflow of 3000 nl/h compared to theoretical auto-correlation of the fractional Gaussian noise

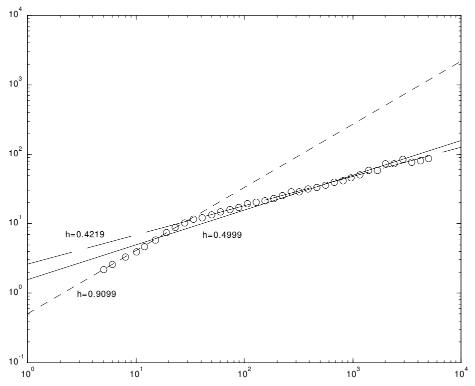


Figure 39: Determination of Hurst parameter with the R/S-statistic for the pressure signal with an airflow of 20000 nl/h

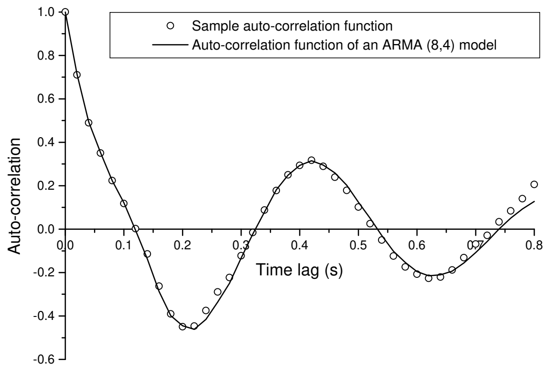


Figure 40: Sample auto-correlation function of an airflow of 14000 nl/ h and the theoretical auto-correlation of the ARMA (8,4) model

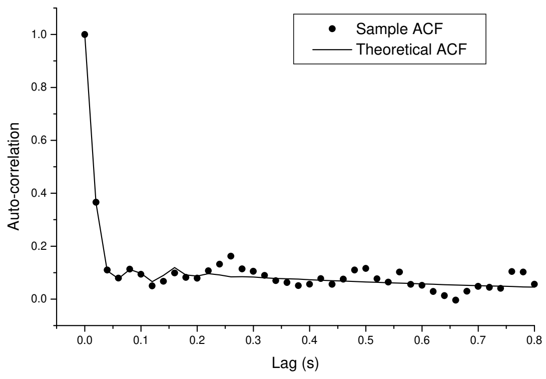


Figure 41: Sample auto-correlation of an airflow of 3000 nl/ h and the theoretical ACF of the ARMA (8,4) model

In Figure 40 the auto-correlation of the pressure signal is shown for an airflow of 14000 nl/h. For higher flow rates it can be seen that the sample auto-correlation does not coincide with the theoretical auto-correlation of the fractional Gaussian noise. The sample auto-correlation can be described well by an ARMA (8,4) model. Interesting to see is the oscilating long term dependence when the air-flow is increased. This indicates that at higher air flows the gas-liquid separator does not work that well and more and more small bubbles are recycled by the downcomer to the riser. As shown, this would not be concluded from the rescaled range method. Therefore it is important to use a sound statistical based method.

But from Figure 41 it appears that the ARMA (8,4) model is not that good to describe the long term dependency for the lower airflow of 3000 nl/h, showing significant deviation for larger timelags. Therefore it would be ideal to combine the features of the two models. The fGN could also be mixed with another model, for example Neogi *et al.* (1993) used the fGn as a random component in conjunction with a sinus, which was fitted to the sample auto-correlation. This because the passage of bubbles will result in a periodic wave component. Another favourite candidate could be the class of *fractional ARMA* models (FARMA), which are able to predict short-time and long-term behaviour.

Therefore the fractional Gaussian noise cannot be used as a model to predict the pressure signal for all air flows, but should be mixed with other models to obtain a model valid for all flow rates and the most promising model will be the fractional ARMA models.

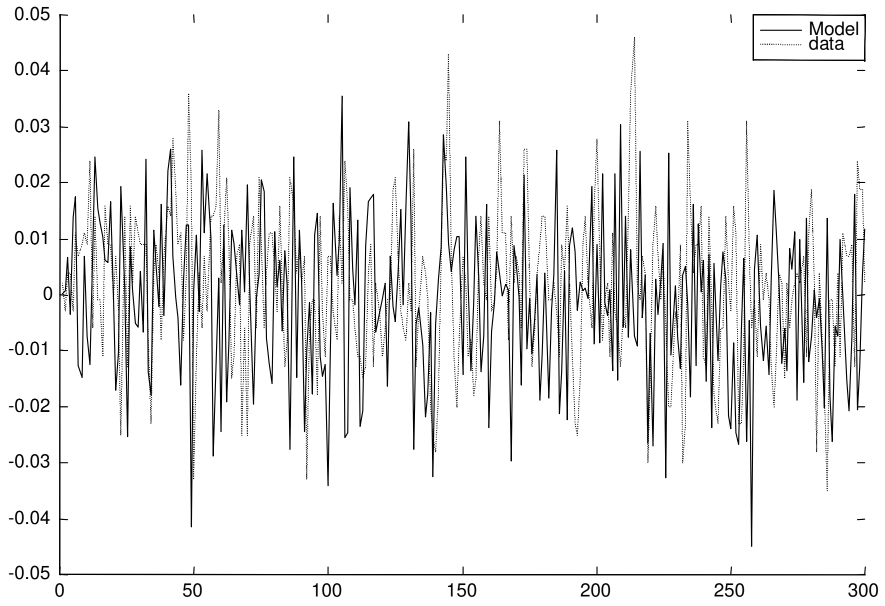


Figure 42: Pressure signal (...) and prediction by the fractional Gaussian noise (-)with H = 0.7218

For the lower air flows fGn can be used as a model or fault diagnositic tool (Scheffer and Maciel, 2001) and is shown in Figure 42. The prediction was done by modifying a simulation method mentioned by Feder (1988):

$$B(t)-B(t-1)=\frac{n^{-H}}{\Gamma\left(H+\frac{1}{2}\right)}\left\{\sum_{i=1}^{n}(i)^{H-1/2}\xi_{(1+n(M+t)-i)}+\sum_{i=1}^{n(M-1)}\left((n+i)^{H-1/2}-(i)^{H-1/2}\right)\xi_{(1+n(M-1+t)-i)}\right\} \quad (4.16)$$

The increment from t to t+1 was calculated by evaluating the two summations in Eq. (4.16). The first summation was calculated by generating ten Gaussian variables with zero mean and a variance of the pressure signal. The second summation in Eq. (4.16) was simply done over the time-series of the pressure signal P. This because the increments in the pressure signal are already a result of evaluating the integral at past times. The calculated increment was added to the past pressure data.

It appears that the pressure signal is predicted reasonable with a mean squared error of 0.0219 and mean absolute error of 0.0174. It should be emphasised that the prediction is for a part biased by the prediction with the generation of several Gaussian variables. Still it can be seen that the FGN gives a satisfactory prediction and thus is a suitable model for measurement monitoring.

It is believed that the prediction could be improved by employing the maximum likelihood of McLeod and Hipel (1994).

Conclusions

It was shown that the fractional Gaussian noise (fGn) is not a suitable model to describe the pressure signal evolution in time for all airflows at a certain height of the air-lift reactor. The fractional Gaussian can be used the model the air-lift reactor for low airflows only. To obtain a model which is valid for all air flow rates, the fGn has to be coupled with other models, which account for the oscillatory behaviour for high air flows. A particular interesting model is the fractional ARMA models which can account for short-term and long-term dependence and is more flexible than the FGN. Therefore modifications have to be done to the fGn to be able to use this model as a fault diagnosis tool or in control applications and preferably the fractional ARMA model should be used.

References

Camarasa, E., E. Carvalho, L.A.C. Meleiro, R. Maciel Filho, A. Domingues, G. Wild, S. Poncin, N. Midoux and J. Bouillard, *Development of a complete model for an air-lift reactor, ISCRE-16, submitted (2000)*

Feder, J. *Fractals*, Plenum Press, New York (1988)

Hurst, H.E., *Long-term storage capacity of reservoirs*, Trans. Am. Soc. Civ. Eng., 116, 770-808 (1951)

Mandelbrot, B.B. e J.W. van Ness, *Fractional Brownian Motions, Fractional Noises and Applications*, SIAM Review, Vol. 10, No. 4, pp. 422-437 (1968)

5. Controle adaptativo, preditivo e não linear com o filtro de Kalman

Resumo – Este trabalho propõe um novo algoritmo preditivo de controle baseado em redes neural restritivas. O algoritmo utiliza um algoritmo seqüencial da programação quadrática para computar a ação seguinte das variáveis manipuladas do processo. O parâmetro preditivo de controle, o fator de supressão é otimizado em linha por um filtro padrão de Kalman. O novo algoritmo de controle foi testado em um processo de fermentação e observou-se que o fator de supressão pode ser identificado, com resultados satisfatórios para ser aplicados em sistemas multivariáveis complexos.

O algorítmo de controle proposto mostrou-se robusto para as situações estudadas sendo, portanto de grande potencial para ser empregado em plantas de grande escala. Particularmente a implementação do algoritmo proposto na forma SISO revela seu potencial para sistemas multivariáveis onde iterações estão presentes sendo necessária a ação do fator de supressão. Pode ser notada a importância dos ajustes deste parâmetro para acomodar as mudanças ocorridas no processo.

Ainda é possível uma melhora no algoritmos de estimação para a correção do fator de supressão que pode determinar com sucesso a taxa de mudança no sistema devido a ação de controle. Isto irá resultar em um algoritmo de controle com respostas mais rápidas e com menores overshoot.

Este capitulo foi publidado nos anais do **ESCAPE -11**.

Computer design of a new predictive adaptive controller coupling neural networks and kalman filter

L. Ender[a], R. Scheffer[b] and R. Maciel Filho[b]

[a]Chemical Engineering Department, Regional University of Blumenau, Rua Antônio da Veiga 140; CP 1507, Blumenau - SC, Brazil, CEP 89010-971

[b]LOPCA/DPQ, Faculty of Chemical Engineering, State University of Campinas (UNICAMP), Cidade Universitária Zeferino Vaz, CP 6066, Campinas - SP, Brazil, CEP 13081-970

This work proposes a new predictive control algorithm based on constraint neural networks. The algorithm utilises a sequential quadratic programming algorithm to compute the next action of the manipulated process variables. The predictive control parameter, the suppression factor, is optimised on-line by a standard Kalman filter. The new control algorithm was tested on a fermentation process and it shows that the suppression factor can be identified and shows promising results to be applied in complex multivariable systems.

INTRODUCTION

Most process control applications consist of not only keeping controlled variables at their set-points but also keeping the process from violating its operating constraints. This is particularly important when there are changes in set-points and when the process is multivariable and non-linear. While good advances have been obtained using conventional predictive controller algorithms, such as DMC, when high performance operation are required, more sophisticated controller are required. In many industrial systems, there exists strong interactions among the process variables as well as internal changes, such as deactivation of a catalyst. To cope with these changes a model predictive control algorithm with self-tuning capabilities is needed.

control algorithm

A predictive algorithm is proposed where the constraints in the manipulated and controlled variables are considered in the on-line minimisation of a quadratic cost criterion. The commonly used minimisation criterion, which compares a reference trajectory, y_{ref}, with the predicted output, y , taking into account the controller movement, is:

$$J = \underset{u}{min\ arg} \sum_{j=1}^{p} \sum_{k=1}^{N_P} \left(y_{ref,j,k} - y_{j,k} \right)^2 + \sum_{j=1}^{m} \lambda_j \sum_{k=1}^{N_C} \left(u_{j,k} - u_{j,k-1} \right)^2 \qquad (1)$$

where p is the number of controlled outputs, m the number of manipulated input variables, N_p the prediction horizon, N_c the controller horizon and λ_j the suppression factor of the corresponding manipulated input variable u_j.

The manipulated inputs and controlled outputs are subjected to the following constraints:

$$y_{min} \leq \hat{y}(k+i) \leq y_{max} \qquad i = 1, \cdots, N \qquad (2)$$

$$u_{min} \leq u(k+i-1) \leq u_{max} \qquad i = 1, \cdots, N_u \qquad (3)$$

$$\left| u(k+i-1) - u(k+i-1) \right| \leq \Delta\ u_{max} \qquad i = 1, \cdots, N_u \qquad (4)$$

The substitution of the classic models used in the prediction along a horizon for a neural network with on-line learning is suggested, due to ability of the nets in representing complex dynamic behaviours, taking into account non-linearity features. The neural network is trained in real time by a closed-loop training scheme as developed by Ender and Maciel (2000)

Adopting the descending horizon technique, only the first control action is implemented and all the calculations are repeated at each sampling time. The minimisation is accomplished by Sequential Quadratic Programming.

The suppression factor λ assures that no exaggerated control action is calculated and influences the systems dynamics. A too small λ results in large control actions, which can result in a instable response, while a too large λ results in a sluggish response. It is common to regard the suppression factor as a design parameter, but this can result in a sub-optimal response if the process is subjected to changes. Normally, the parameter is tuned manually until a desired process behaviour is obtained, which can be a very time-consuming procedure. Additionally, different values of λ might be needed for other operating conditions.

Therefore it is proposed to create an automatic estimation procedure for λ to ensure an optimal value of λ, which will result in a satisfactory control for all process situations.

The neural network

For the development of this work the feedforward architecture with back-propagation learning and sigmoidal function were used. In all the used neural network configurations, one hidden layer was considered, because Hiecht-Nielsen (1989) proved that any continuous function can be approached for any degree of precision using a backpropagation neural network with three layers, if there is sufficient number of active neurons in the hidden layer.

The patterns presented to the neural network are composed of information from the last inputs/outputs and the current disturbance. This neural network when used in the prediction becomes recurrent. In this case, the weights are adjusted on-line, at the same time in such a way that it is used in the predictions.

Estimation of the suppression factor

The adjustment or tuning algorithm of the parameter λ is based on the standard Kalman filter. To be able to adjust λ a dynamical system has to be created which can observe the state of the parameter λ as in:

$$\begin{aligned} \lambda_{k+1} &= \lambda_k \left(+ w_k\right) \\ z_k &= C\,\lambda_k \left(+ v_k\right) \end{aligned} \tag{5}$$

where w_k and v_k are random variables with a normal distribution of N(0,Q) and N(0,R) respectively. z_k is the measurement related to the state λ_k. Normally the noise of a parameter state is zero, but a small process noise results in a more stable filter.

The observation equation of the λ can be derived from the minimisation criterion J in equation 1 assuming that the SQP minimisation was successful. The first derivative of J will be zero, when J is in its minimum. It is assumed that only u_k, the implemented control action, is of importance. The last input, u_{k-1}, is regarded as a constant This can be validated because the receding horizon technique implements the first control action only. This results in the following dynamical system:

$$\begin{aligned} &\lambda_{k+1} = \lambda_k + w_k \\ &\sum_{i=1}^{q} \left\{ \frac{\partial y_i}{\partial u_k} \left(y_{ref,i} - y_i \right) \right\} = \left(u_k - u_{k-1} \right) \lambda_k + v_k \end{aligned} \tag{6}$$

The measurement, z_k, in equation 6 is represented by a summation of derivatives multiplied by the error of the process output with the output reference. The estimation of the measurement is done by the right side with $E\{v_k\}=0$. v_k is chosen as random variable with a normal distribution of (0,1). As the measurement is actually an mathematical expression, the variance of the measurement noise is chosen 1. This is mainly done to prevent a divide by zero in the Kalman filter, which occurs when u_k equals u_{k-1}.

It was already said that for a parameter the process noise w_k would be discarded, but it can be shown that the Kalman filter converges to a constant gain matrix for linear systems (Goodwin and Sin, 1984). λ is wanted to change in real-time to cope with changing process conditions and an elegant way to do so is to make the process noise a function of an error. It is proposed to make the variance of the process noise a function of error between the reference and prediction of the process output by the neural network:

$$w_k \propto \left(0, \left(y_{ref,i} - y_{ANN,i}\right)^2\right) \tag{7}$$

In this way the uncertainty of the parameter will be high if the process is far away from its set-point and thus the Kalman filter will be triggered, resulting in a larger adjustment of the parameter λ by the measurement.

Results

A large industrial fermentation process for the production of penicillin is considered as a case-study. The simulation is based on a model of Rodrigues (1996), which is validated with industrial data. The process is a fed-batch process and falls down in two parts, the growing phase and the production phase. In the growing phase a high sugar level is maintained, while in the production phase it has to be kept low as it inhibits the penicillin production.

The emphasis is put on the production phase where the feeding strategy of the sugar substrate has to be chosen carefully to maximise the penicillin production.

One input, the substrate feed flow, and four state variables, the bio-mass concentration, the substrate concentration, the penicillin concentration and the dissolved oxygen concentration, were taken to identify the process.

The substrate feed influences directly the cellular concentration, the substrate concentration and the penicillin concentration. The dissolved oxygen concentration is also influenced, because of the cellular growth which requires oxygen for respiration.

The dissolved oxygen concentration is a vital variable for a good process operation. Rodrigues (1996) mentioned that a dissolved oxygen concentration lower than 30% causes deterioration of the fungi. The dissolved oxygen concentration can be controlled by aeration and stirring, but the latter will destroy the fungi at high rotation speeds. This shows the need for an optimisation algorithm which takes into account such constraints.

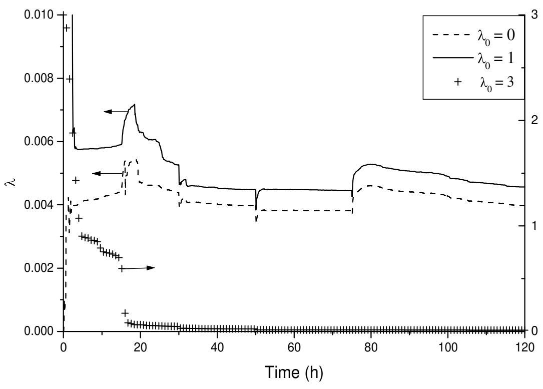
A single input, single output (SISO) system was set-up to verify the proposed process control scheme. The controlled variable was the dissolved oxygen concentration which was controlled by the rotation speed. Various constraints are applicable here as, maintaining the dissolved oxygen concentration above 30% and avoiding high rotation speeds which destroy the fungi. A white noise was imposed on the dissolved oxygen concentration to simulate measurement noise.

It was started with a manual tuning period to determine the stability of the process. Unfortunately, the penicillin production process has a very slow dynamics, for which λ resulted to be zero. Note that this is valid for most SISO control problems where the manipulated variable can change at its maximum rate. In the SISO case the proposed control strategy becomes therefore interesting for very sensible and unstable systems only. Still it can be interesting to see if the proposed control scheme is stable and to see if a value different of zero is found or actually that a value near zero is encountered.

λ has to be greater or equal to zero, and it can be seen from equation 6 that this will not be always the case. Therefore the dynamic system 6 was modified by taking the absolute values of both sides, which will result in positive values for λ. Another possibility is taking the positive value of the resulting lapda. Both strategies were tested

In Fig. 1 it is shown the convergence of parameter λ for various initial values, it can be seen that the parameters converge sooner or later to the same value and from then on show the same dynamic behaviour. It can be seen from Fig. 2 that there is only a minor difference in the beginning for the dissolved oxygen concentration and it looks like that the control is better when the initial λ is different from zero but not to far away from its optimal value otherwise it constrains the input action to much.

It can be seen that with measurement noise present that the parameter λ does not stay constant, which is due to the insertion of process noise into the dynamic estimation equation of the parameter λ. But it can be seen that this is also due to the developed identification system. After a set-point change the controller movement will become rapidly close to zero while the measurement calculation will always be different from zero due to the measurement noise, resulting in small changes of λ. This will be problem when the absolute values are



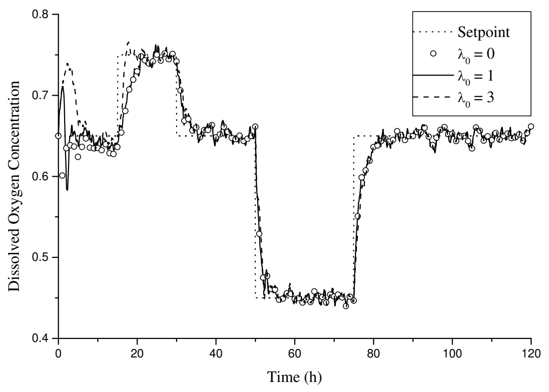
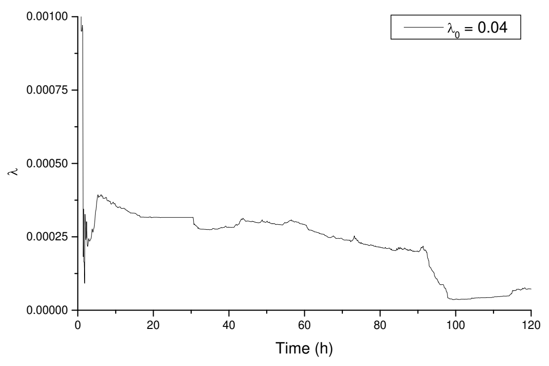


Fig. 1: Estimation of suppression factor for different initial values

Fig. 2: Controlled dissolved oxygen concentration for the servo problem

implemented, because this results in a parameter drift until a new set-point change or perturbation occurs.

In Fig. 3 and Fig. 4 the control system is shown for a complex regulation problem, where a series of step changes is implemented which maximise the penicillin production. The sugar concentration affects the dissolved oxygen concentration due to the cellular growth and due to the energy required in the metabolism for the penicillin manufacturing. Even with a series of perturbations of every 6 and 6 hours the penicillin process is maintained under satisfactory control with an maximum error of 5% after the tuning of the suppression factor. It can be seen that the parameter λ is never constant, but as in the servo case never diverges and never shows instability.



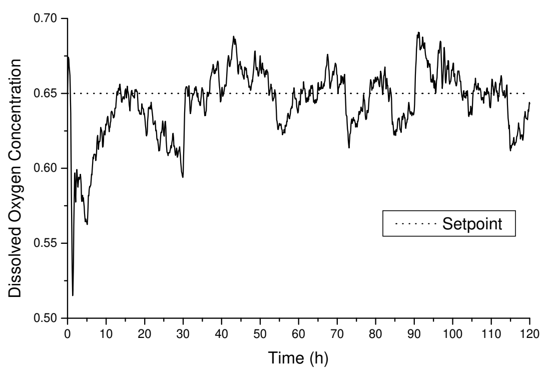


Fig. 3: Evolution of the suppression factor for the regulator problem

Fig. 4: Controlled dissolved oxygen concentration for the regulator problem, perturbation in the substrate feedstream

A test was done with no measurement noise present to test the behaviour of λ and to see if it would settle on a constant value or settle to zero. In Fig. 5 a comparison is made between the proposed estimation system with the self-tuning method of λ and a fixed λ.

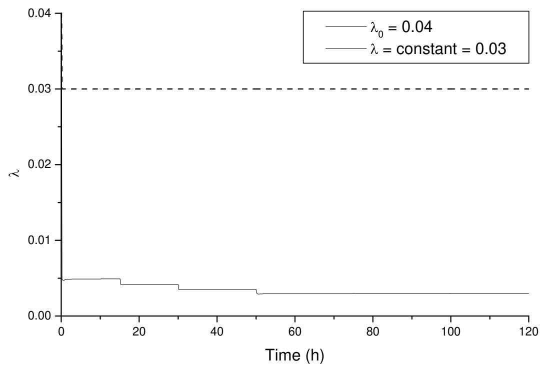


Fig. 5: Suppression factors without measurement noise

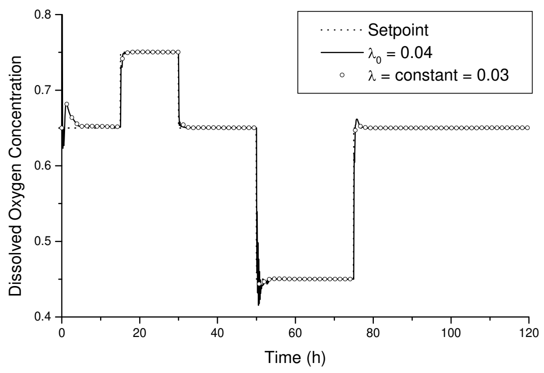


Fig. 6: Control of dissolved oxygen concentration without measurement noise

First it can be seen from the dissolved oxygen response (Fig. 6) with a fixed λ that there is actually a need for variable λ. At some set-point changes the dissolved oxygen concentration is controlled vary well, while at other set-point changes the same value of λ results in a small overshoot.

The proposed tuning system is still able to tune the suppression factor but does not prevent the small overshoots, which appear with the constant suppression factor. Therefore the developed identification method of λ has to be modified to successfully identify the possible hazard of a overshoot.

conclusion

The proposed control algorithm has shown to be robust for the analysed disturbances, promising to have a great potential to be used in control strategies of large scale systems. Particularly, the application of the proposed algorithm in the SISO case reveals its potential for multivariable systems where the interactions and thus the suppression factors are more important. It is noted the importance of the adjustment of the suppression factor to be able to cope with change in process operations.

Still there has to be searched for a better estimation algorithm of the suppression factor, which can determine successfully the rate of change of the system due to the control action. This will result in a fast control algorithm with little or no overshoot.

ACKNOWLEDGEMENT

The authors would like to thank Capes for the financial support in the form of a Scholarship.

REFERENCES

Ender, L. and R. Maciel Filho, Brazilian Journal of Chemical Engineering (2000),no, vol, pg, "Design of multivariable controller aided by computer based on neural networks"

Hecht-Nielsen, R. IEEE Int. Conf. On Neural Networks San Diego (1989) v.I, p.593-605, "Theory of the Backpropagation Neural Network"

Goodwin G.C. and K.S. Sin, (1984) Adaptive filtering, Prediction and control, Prentice-Hall, Englewood Cliffs, NJ

Rodrigues, J.A.D., (1996), "Dinâmica e estratégias de otimização e controle do process de penicilina", PhD thesis, State University of Campinas

6. Conclusões e trabalho futuro

Vários aspectos de um sistema de controle avançado foram estudados com o enfoco em identificação de sistemas não-lineares, monitoração da medida, ação de controle e a otimização não-linear com restrições.

O filtro de Kalman foi aplicado em três dos quatro itens mencionados: a identificação dos sistemas não lineares em tempo real por redes neuronais treinados com o filtro de Kalman, o controlador de variância mínima, que calcula o passo de controle por filtro de Kalman e em otimização não linear com restrições.

Demonstrou-se que a aplicação do filtro de Kalman resulta em uma convergência global melhor em menos iterações. Em treinamento em tempo real, encontrou-se que o filtro de Kalman resulta em uma predição em tempo real boa com uma aprendizagem das características do modelo melhor do que o algoritmo de back-propagacão padrão. No caso industrial, o algoritmo funcionou sucedido sob grande erro da medida. Não foi encontrado uma grande melhora pela aplicação de matriz de erro do processo em função de erro. Na otimização não linear com restrições foram obtidos resultados muitos promissórios para obter um algoritmo de convergência global com poucas iterações. Não se encontrou divergência para controle com uma rede neural que nunca foi treinado antes. Em algumas casos foi necessário de perturbar o matriz de Hessiano, mas mesmo assim foi obtido em quase todos os casos o ótimo global. Não houve divergência mesmo no controle em tempo real com uma rede neural que nunca foi treinado antes. Porém, a força computacional pode ser alto demais para problemas de larga escala e pode diminuir a capacidade da aplicação dos algoritmos desenvolvidos.

Por causa disto é recomendado de utilizar o filtro the “unscented Kalman” no treinamento em tempo real e otimização não linear com restrições. Especialmente no ultimo caso, haverá a possibilidade de obter-se um algoritmo de otimização não linear com restrições rápido e com propriedades de convergência global.

O método da monitorização da medida foi visto pelo ruído Gaussiano fracionário, que pode ser utilizado para descrever sinais complexos. Os sinais complexos podem ser descritos parcialmente por o ruído Gaussiano fracionário, mas o modelo é bastante inflexível e portanto não pode ser utilizado como um modelo geral para monitorização de sinal de medida.

Por isto é recomendado de desenvolver os modelos ARIMA fractionarios como modelos de monitorização, por causa que estes modelos são capazes de descrever dependência de curto e longo prazo e então, serão muito bons modelos para uma ferramenta geral para monitorização do sinal de medida.

7. Referências bibliográficas

Astrom, K.J. and B. Wittenmark (1995), Adaptive Control, 2ed., Addison-Wesley Publishing Company

Bertsekas, D.P. (1994), Proc. of the 33rd conf. on decision and control, Lake Buena Vista, Fl, “Incremental least squares methods and the extended Kalman filter”

Bierman, G.J. (1976), Automatica, v. 12, n. 4, pp. 375-382, U-D factorization

Camarasa, E., E. Carvalho, L.A.C. Meleiro, R. Maciel Filho, A. Domingues, G. Wild, S. Poncin, N. Midoux and J. Bouillard, (2001), Chem. Eng. Science, v. 56, pp. 493-502 "Development of a complete model for an air-lift reactor"

Crueger, W. and A. Crueger, (1984) Biotechnology: a Textbook of Industrial Microbiology, (Suderland, Sinawer Associates, Inc.

Edgar, T.F. and D.M. Himmelblau, (1989) McGraw-Hill Inc., “Optimisation of chemical processes”

Doucet, A., N. de Freitas and N. Gordon (2001), “Sequential Monte Carlo Methods in Practice”, Springer-Verlag

Drahos, J. F. Bradka and M.Puncochar (1992), Chem. Eng. Science, v. 47, n. 15/16, pp. 4069-4075, “Fractal Behaviour of Pressure Fluctuations in a Bubble Column”

Fan, L.T., D. Neogi, M.Yashima and R.Nasser, (1990), v. 36, n. 10, pp. 1529-1535, “Stochastic Analysis of a Three-Phase Fluidized Bed: Fractal Approach”

Feder, J. (1988), Fractals, Plenum Press, New York

Franca, F., M. Acikgoz, R.T. Lahey Jr and A. Clausse (1991), Int. J. Multiphase Flow, v. 17, n. 4, pp. 545-552, “The Use of Fractal Techniques for Flow Regime Identification”

Gordon, N.J., D.J. Salmond and A.F.M. Smith (1993), IEE Proceedings-F v. 140, n. 2, pp 1070113, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”

Haykin, S. (1999), Prentice-Hall Inc., New Jersey, “Neural networks: a comprehensive foundation”

Heimes, F. (1998), Proc. of int. the 1998 IEEE conf. on systems, man and cybernetics, Orlando, pp 1639-1644, “Extended Kalman filter neural network training: experimental results and algorithm improvemens”

Hipel, K.W. and A.I. McLeod (Elsevier, 1994), "Time Series Modelling of Water Resources and Environmental Systems"

Hurst, H.E., (1951), *Trans. Am. Soc. Civ. Eng., 116, 770-808*, "Long-term storage capacity of reservoirs"

Julier, S. and J.K. Uhlmann (1994), Internet Publication: http:/www.robots.ox.ac.uk/~siju/index.html, Technical report, Robotics Research Group, Department of engineering Science, University of Oxford, “A general method for approximating nonlinear transformations of probability distributions”

Julier, S. and J.K. Uhlmann (1997), Proc. of aerosSense:11th int. symp. On aerospace/defence sensing, simulation and controls, “A new extension of the Kalman filter to nonlinear systems”

Kalman, R.E. (1960) Journal of basic engineering, Transactions of ASME, series D, v. 82, n. 1, pp 35-45, “A new approach to linear filtering and prediction problems”

Kailath, T. (1974) IEEE trans. on information theory, v IT-20, n 2, pp 146-181, “A view of three decades of filtering theory”

Lewis, F.L. (1986), "Optimal Estimation", John Wiley & Sons, New York, NY

Mandelbrot, B.B. and J.W. Van Ness, *SIAM Rev.* (1968), 10(4), 422-437, "Fractional Brownian Motions, Fractional Noises and Applications"

McLeod, A.I. and K.W. Hipel, (1978) *Water Resources Research*, 14 (3), 491-508, "Preservation of the Reascaled Adjusted Range - 1. A Reassessment of the Hurst Phenomenon"

Meinhold, R.J. and N.D. Singpurwalla (1983), American Statistician v. 37, n. 2, Understanding the Kalman filter

Merwe, R. van der and E.A. Wan (2001), European symposium on artivial neural networks (ESANN), Brugge, Belgium, "Efficient derivative-free Kalman filters for online learning"

Messina, G., L. Lorenzoni, O. Cappeliazzo, A. Gamba, (1983), La Chimica e l´Industria, v. 65, n.1,pp 10-17 , "Side reactions and related by-products in the phenol/acetone process"

Morf, M. and Kailath, T. (1975), IEEE Trans. Automatic Control, AC-20, pp 487-497, Square-root algorithms for least squares estimation.

Narendra, K. S. and K. Parthasaraty (1991), IEEE trans. on neural networks, v 2, n 2, pp 252-262, "Gradient methods for the optimization of dynamical systems containing neural networks"

Neogi D., LT Fan, Y. Kang and M. Yashima (1993), Aiche Journal, v. 39, n. 3, pp 513-517, Fractal analysis of fluidized particle behaviour in liquid solid fluidized beds

Nielsen, J.N. and H. Madsen (2001), Automatica, v. 37, n. 1, pp. 107-112, Applying the EKF to stochastic differential equations with level effects

Norgaard, M., N.K. Poulsen and O.Ravn (2000), Automatica, v 36, pp 1627-1638, "New development in state estimation for nonlinear systems"

Palmieri, F., M. Datum, A. Shah and A. Moiseff (1991), Proc. of int. joint conf. on neural networks, Seattle, Wa, pp I-125-131, "Sound localization with a neural network trained with the multiple extended Kalman algorithm"

Papoulis, A. (1991), "Probability, random variables, and stochastic processes", 3rd ed., McGraw-Hill

Puskorius, G.V. and L.A. Feldkamp (1991), Proc. of int. joint conf. on neural networks, Seattle, Wa, "Decoupled extended Kalman filter training of feedforward layered networks"

Puskorius, G.V and L.A. Feldkamp (1994a), IEEE trans. on neural networks, v 5, n 2, pp 279-297, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks"

Puskorius, G.V and L.A. Feldkamp (1994b) IEEE , pp 2488-2493, “Truncated backpropagation through time and Kalman filter training for neurocontrol”

Puskorius, G.V. and L.A. Feldkamp (1997), Int. conf. on neural networks, ICNN’97, vol 3, pp 1879-1883, “Extensions and enhancements of decoupled extended Kalman filter training”

Puskorius, G.V. and L.A. Feldkamp (1999), Proc. of int. joint conf. on neural networks ’99, Washington, D.C., “Roles of learning rates, artificial process noise and square root filtering for extended Kalman filter trianing”

Rivals, I. and L. Personnaz (1998), Neurocomputing, 20(1-3), 279-294, “A recursive algorithm based on the extended Kalman filter for the training of feedforward neural networks”.

Rijnsdorp, J.E., *Kwaliteitsbeheersing en receptaanpassing in de procesindustrie*, Proces Technologie, 11, 41-44 (1993)

Rodrigues, J.A.D and R. Maciel Filho (1999), Chemical Engineering Science, 54(13-14), 2745-2751, "Production optimisation with operating constraints for a fed-batch reactor with DMC predictive control "

Rumelhart, D.E., E. Hinton and P.J. Williams (1986) MIT press, in “Parallel distributed processing", v 1, ed. D.E. Rumelhart and J.L. McClelland, "Learning internal representation by error propagation”

Ryoo, H.S. and N.V Sahindis (1995), Comp. and Chem. Engng. 19(5), 551-566, "Global optimization of nonconvex NLPs and MINLPs with applications in process design"

Saad, E.W., D.V, Prokhorov, D.C. Wunsch (1998), IEEE Transactions on neural networks, Comparitive study of stock trend prediction using time delay recurrent and probabilistic neural networks

Scheffer, R. & R. Maciel Filho (2000), ‘Training a Recurrent Neural Network by the Extended Kalman Filter as an Identification Tool’, Escape-10 symposium proceedings, pp 223-228

Scheffer, R. and R. Maciel Filho *ISCRE-16 Poland to be published in Chemical Engineering Science* (2001a), "The fractional Brownian motion as a model for an industrial air-lift reactor"

Scheffer, R. & R. Maciel Filho (2001b), ‘process identification of a fed-batch penicillin production process - training with the extended kalman filter’ (In: "Application of Neural Network and Other Learning Technologies in Process Engineering ", Ed. I.M.Mujtaba and M.A. Hussain, Imperial College Press

Shah, S. and F. Palmieri (1990), Proc. of int. joint conf. on neural networks, San Diego, Ca, June, pp III-41-45, “MEKA - A fast, local algorithm for training feedforward neural networks”

Shah, S., F. Palmieri and M. Datum (1992), Neural networks, v 5, pp 779-787, “Optimal filtering algorithms for fast learning in feedforward neural networks”

Singhal, S. and L. Wu (1989), Proc. of IEEE int. conf. on acoustics speech and signal processing, Glasgow, Scotland, pp 1187-1190, “Training feed-forward networks with the extended Kalman filter”

Söderström, T., (1994), "Discrete-time stochastic systems: estimation and control", Prentice Hall International (UK) Limited

Tvrzská de Gouvêa, M. and D. Odloak (1998), Computers and chem. engng, v 22, n 11, pp 1623-1651, “A new treatment of inconsistent quadratic programs in a SQP-based algorithm”

Wan, E.A. and R. van der Merwe (2000) Proc. of symposium 2000 on adaptive systems for signal processing, communication adn control (AS-SPCC), Lake Louise, Alberta, Canada, pp 153-159, “The unscented Kalman filter for nonlinear estimation”

Werbos, P.J. (1974) PhD thesis, Harvard university, Cambridge, MA, “Beyond regression: New tools for prediction and analysis in the behavioral sciences”

Anexo 1 - The (Extended) Kalman Filter

Abstract – The Kalman filter takes a central place in most of the works developed in this thesis, and therefore a preliminary discussion of the mathematical and statistical axioms is needed to understand the work. The filter equations are reviewed from the "deterministic" and the "statistic" viewpoints. The deterministic approach reveals the filter equations easily, while the statistic approach points out better the improvements which can be made. Its characteristics and behaviour will be discussed and some recent insights from literature, which can be used in future implementations. The different ways in which the Kalman filter can be utilised are state and parameter estimation, neural network training algorithm (parameter estimation), controller and constraint optimisation algorithm.

Introduction

The Kalman filter is part of the family of the least squares algorithms, which were apparently introduced by Gauss in 1795 but first published by Legendre in 1805 (Kailath, 1974). Even the recursive algorithm seems not be that recent, because Gauss was forced to invent it to help astronomers determine the orbit of the asteroid Ceres. The Kalman filter incorporates all other recursive least squares algorithms such as the known recursive least square algorithm.

Before Kalman (Figure 43) published his famous paper (Kalman, 1960), the filtering problems were mainly approached from the frequency domain, utilizing the auto and covariance functions, such as in the celebrated Wiener filter. The replacement of these functions by state space models led to a flexible way of estimation and filtering.

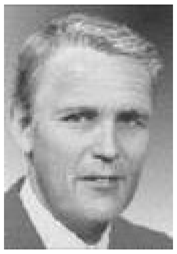


Figure 43: Rudolf E. Kalman

The reason of the Kalman filters popularity is that it is the optimal linear filter in case of Gaussian process and measurement noises. <u>If they are not Gaussian then the Kalman filter will still be the best linear filter possible</u>. In the non-linear case the filter will be sub-optimal, due to a linearization of the non-linear equations in every prediction and correction step. In this case the Kalman filter is denoted as the extended Kalman filter (EKF), but in fact the Kalman filter equations are the same. The EKF is a popular method as it is less laborious than non-linear state observers which are system dependent.

The unscented Kalman filter proposed by Julier and Uhlman (1994 and 1997) is an improvement of the extended Kalman filter. The unscented Kalman filter obtains a better approximation of the post-error covariance distribution, besides the advantage that it is a derivative-free technique. The CDF filter proposed by Norgaard, Poulsen and Ravn (2000) is based on a central difference interpolation scheme of Stirling and obtains a better approximation of the post-error covariance distribution also. It was shown by Wan and van der Merwe (2001) that both approaches can be unified in one general scheme.

Still, after Kalman's discovery the filter became quite unpopular as it was said to diverge. This is mainly due the computer implementation, if done wrong, round off errors lead to a non-positive definite error covariance matrix. Several formulations of the Kalman filter can prevent this divergence such as the *Joseph stabilised version* of the measurement error covariance update (Lewis, 1986), square root algorithms (Morf and Kailath, 1975) and the U-D factorisation (Bierman, 1976). The U-D factorisation is actually a square root algorithm also. The price of these stable Kalman filter algorithms is an increased computational load.

Statistical background

A very good and didactic book on probability and stochastic processes was written by Papoulis (1991). The book of Söderström (1994) has a very good summary of the statistics needed for Kalman filtering, but to the authors' opinion, the best book written on estimation and statistics for Kalman filtering was written by Lewis (1986).

The probability theory was introduced by Kolmogorov and concerns the study of probability spaces and random variables. The latter is introduced in paragraph 0. A probability space is defined by a triple (Ω, S, P):

- Ω is often called the sample space which contains the empty set ∅ and all the possible outcomes of a random experiment
- S is a σ-algebra of the sample space and its members are events. A σ-algebra means that it contains Ω, it complements of its events are events and that the union of a finite sequence of events is also a event. If Ω is discrete then S and Ω are the same.
- P is the probability measure on Ω, such that its P(Ω) = 1

Various events, such as A, B and C or car, apple and bicycle form a set. Sets can be submitted to various operations, such as sum or union, product or intersection and complement, which are easily visualised in Venn diagrams (Figure 44).

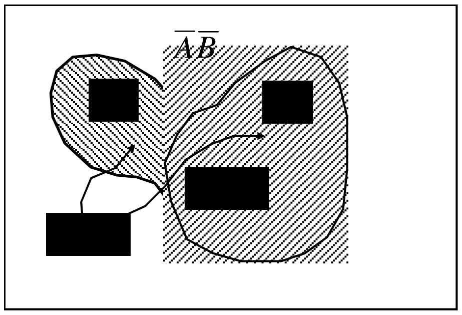


Figure 44: Venn Diagram of Sets A and B with the operations union (A∪B or A+B), intersection (A∩B or AB) and complement ($\overline{AB}$)

The events A, B, $\overline{AB}$ and ∅ (the empty set) form the event space.

From the Venn diagram it is now easily understood that the sum of the probabilities of event A and B is given by:

$$P(A+B)=P(A)+P(B)-P(A\cap B) \tag{7.1}$$

If the events are independent, the intersection of event A and B is given by:

$$P(A\cap B)=P(A)\cdot P(B) \tag{7.2}$$

The conditional probability of B given A is defined as:

$$P(B/A) = \frac{P(A \cap B)}{P(A)} \tag{7.3}$$

which is the probability of event B, when we know that event A happened. For example, if we throw a die the probability to obtain a 2 is 1/6. But if we have the additional knowledge that the event which happened was even then probability becomes 1/3, because the event 2 is one of the possible three even outcomes of the experiment.

The event A could be divided in various sub-events A_i. The event A is divided in 9 subsets in Figure 45.

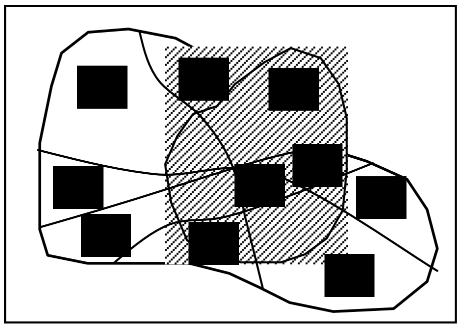


Figure 45: Venn Diagram of Sets A and B where set A is divided in 9 subsets

The conditional probability of event A_i given event B is given by *the Bayes theorem.* The index n is 9 for the partition in Figure 45.

$$P(A_i/B) = \frac{P(B/A_i)P(A_i)}{P(B/A_1)P(A_1) + \cdots + P(B/A_n)P(A_n)} \tag{7.4}$$

The probability of $P(A_i)$ is often called the *a priori* distribution, while the probability of $p(A_i|B)$ is called the *a posteriori* distribution. These concepts are used in Bayesian estimation and are used in the statistical definition of the Kalman filter.

Random variable

A random variable (rv), ***x***, is a function which relates one of the possible outcomes from the event space S to a number from the real domain $\Re$. This random variable is defined by the triple (S, F, P). S is the set of all possible outcomes which have a probability measure P limited to the value of zero for the impossible event and 1 for the set S. F is a Borel Field, which contains all possible sums and intersections of the possible outcomes and the empty set.

If the event is non-countable then it is defined a probability on the real line for the event $P(x=x_i)$ that can be calculated by the integral:

$$F(x_i) = P(x \le x_i) = \int_{-\infty}^{x_i} f(x) \cdot dx \tag{7.5}$$

f(x) is the probability density function on x, which has to satisfy the property:

$$\int_{-\infty}^{\infty} f(x) \cdot dx = 1 \tag{7.6}$$

The expectation of the random variable ***x*** is denoted by E{***x***}.

$$E\{x\} = \int_{-\infty}^{\infty} x \cdot f(x) \cdot dx \tag{7.7}$$

This quantity is normally known as the mean and would be for a dice equal to 3.5.

The mean is the first moment of the random variable, higher moments are calculated by:

$$E\{x^n\} = \int_{-\infty}^{\infty} x^n \cdot f(x) \cdot dx \tag{7.8}$$

Instead of the second moment, it is used the variance of the random variable **x**, which is defined by:

$$E\{(x - E\{x\})^2\} = \int_{-\infty}^{\infty} (x - E\{x\})^2 \cdot f(x) \cdot dx \tag{7.9}$$

A random variable can suffer a (non) linear transformation resulting in a other random variable y.

$$y = g(x) \tag{7.10}$$

The probability density function of rv y can be calculated from the distribution of rv x

$$f_y(y) = \sum_{i=1}^{number\,of\;roots} \frac{f_y(x_i)}{|g'(x_i)|} \tag{7.11}$$

where x_i is the ith root of the equation g(x) = y. Thus for the equation y = Ax +B, the only root is x = (y-B)/A.

Luckily, the expectation of y can be calculated directly from the probability density function of rv x

$$E\{y\} = E\{g(x)\} = \int_{-\infty}^{\infty} g(x) \cdot f_x(x) \cdot dx \tag{7.12}$$

Given two random variables x and y and a function g(x,y), then their joint expectation is given by:

$$E\{g(x,y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y) \cdot f_{xy}(x,y) \cdot dx \cdot dy \tag{7.13}$$

Instead of the joint distribution it can be used the conditional distribution of x given y.

$$E\{g(x,y)\} = E_y\{E_{x/y}\{x,y\}\} = \int_{-\infty}^{\infty} y \cdot \left(\int_{-\infty}^{\infty} g(x,y) \cdot f_{x/y}(x,y) \cdot dx \right) \cdot f_y(y) \cdot dy \tag{7.14}$$

The continuous version of the conditional distribution of Eq. (7.3) is given by:

$$f_{x/y}(x,y) = \frac{f_{xy}(x,y))}{f_y(y)} \tag{7.15}$$

The Bayes theorem of Eq. (7.4) becomes for the continuous case:

$$f_{x/y}(x/y) = \frac{f(y/x)) \cdot f_x(x)}{f_y(y)} \tag{7.16}$$

Estimation Techniques

Having defined the most important statistical properties it is introduced the following estimation techniques, which are used in filtering, state and parameter estimation.

- Mean Square estimation
- Linear Mean Square Estimation

All estimation techniques minimises a quadratic cost function of difference between the observed value and the predicted value by a function.

In Mean Square Estimation it is minimized the cost-function.

$$J = E\left\{(y - c(x))^2\right\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(y - c(x))^2 \cdot f_{xy}(x, y) \cdot dx \cdot dy \tag{7.17}$$

Utilising the definition of the conditional distribution this can be rewritten as:

$$J = E\left\{(y - c(x))^2\right\} = \int_{-\infty}^{\infty} f(x) \cdot \left(\int_{-\infty}^{\infty}(y - c(x))^2 \cdot f_{y/x}(x, y) \cdot dy \right) \cdot dx \tag{7.18}$$

By setting the derivative of the cost-function to c(x) equal to zero, the minimum square estimate is obtained, which is:

$$c(x) = E\{y/x\} = \int_{-\infty}^{\infty} y \cdot f_{y/x}(x, y) \cdot dy \tag{7.19}$$

Normally, the conditional distribution is not known except for Gaussian distributions. This conditional distribution of y|x can be found by defining a vector of $[x\ y]^T$ and fill it in the joint normal distribution. With the definition of the conditional distribution it is found the expressions of the conditional mean and the conditional covariance as the estimate and the error of the estimate.

The mean square estimation is also denoted as Bayesian estimation and is used in explaining the Kalman filter for statisticians by Meinhold and Singpurwalla (1983). The conditional distribution of Eq. (7.19) could be viewed in the following way by filling in Bayes theorem of Eq. (7.16) in Eq. (7.14).

$$P\{state/data\} \propto P\{data/state\} \cdot P\{state\} \tag{7.20}$$

In Linear Mean Estimation it is assumed that the conditional expectation is a linear function of the measurement. The method is reviewed in paragraph 0.

Thus the difference between the mean square estimate and the linear mean square estimate is in the supposition done. In mean square estimation it is needed the conditional distribution, which has solution only for Gaussian variables, while in the linear mean square estimation the conditional distribution is supposed to be linear.

Deterministic derivation

As said one of the basic assumptions of the Kalman filter is the state space model, a general state space model can be described by:

$$x_{k+1} = Ax_k + Bu_k + Gw_k \tag{7.21}$$

$$y_k = Cx_k + v_k \tag{7.22}$$

w_k and v_k are random Gaussian variables with a distribution of N(0, $\sqrt{Q}$) and N(0, $\sqrt{R}$) respectively. u_k is a deterministic input. Normally the noises and the states are said to be uncorrelated, which means that the expectations E[xw] and E[wv] are 0. The expectation of x, E[x], is the mean of x and the expectation of E{[x-E(x)][x-E(x)]^T} is called the variance of x, whose square root equals the standard deviation for Gaussian distributed variables.

A typical example of x would be a concentration which is measured by a temperature. This occurs often in distillation columns, where the concentration can be determined by the temperature through a thermodynamic model.

If no measurement is available then the best estimate of x is given by:

$$\hat{x}_{k+1}^- = E\{x_{k+1}^-\} = A\hat{x}_k^- + Bu_k \tag{7.23}$$

as the estimate of E{w}=0.

The initial estimate, x_0, of x is the mean of x. The minus means that the estimate (indicated by ^) is done before the measurement, which is also known as the time update or prior estimate.

The core of filtering problems is the error covariance matrix, P, with the error defined as the real x minus its estimate and is for the system in Eq. (7.21) given by

$$P_{k+1}^{-} = E\left\{\left(x_{k+1} - x_{k+1}^{-}\right)\left(x_{k+1} - x_{k+1}^{-}\right)^T\right\} = AP_k^{-}A^T + GQG^T \tag{7.24}$$

which is the change in the estimation of the error through the dynamics of the system. The process noise is sometimes viewed to be originated from the inputs, in that case G should be replaced by B. The inverse of the error covariance matrix P is called the information matrix.

Suppose now that a measurement y is available, then it would be possible to improve our estimate of the state variable by a error correction of the state:

$$\hat{x}_k^{+} = \hat{x}_k^{-} + K\left(y_k - \hat{y}_k\right) = \hat{x}_k^{-} + K\left(C\left(x_k - \hat{x}_k^{-}\right) + v_k\right) \tag{7.25}$$

where K is denoted as the Kalman gain. What to do to determine the Kalman gain? Lets calculate the error covariance matrix after the measurement.

$$\begin{aligned}
P_k^{+} &= E\left\{\left(x_k - \hat{x}_k^{+}\right)\left(x_k - \hat{x}_k^{+}\right)^T\right\} \\
&= E\left\{\left(x_k - \hat{x}_k^{-} - K\left(C\left(x_k - \hat{x}_k^{-}\right) + v_k\right)\right)\left(x_k - \hat{x}_k^{-} - K\left(C\left(x_k - \hat{x}_k^{-}\right) + v_k\right)\right)^T\right\} \\
&= E\left\{\left(x_k - \hat{x}_k^{-}\right)\left(x_k - \hat{x}_k^{-}\right)^T\right\} - 2E\left\{\left(x_k - \hat{x}_k^{-}\right)\left(x_k - \hat{x}_k^{-}\right)^T\right\}C^T K^T + \\
&\quad K\left(CE\left\{\left(x_k - \hat{x}_k^{-}\right)\left(x_k - \hat{x}_k^{-}\right)^T\right\}C^T K^T + KE\left\{v_k v_k^T\right\}K^T\right) \\
&= \left(I - KC\right)P_k^{-}\left(I - KC\right)^T + KRK^T
\end{aligned} \tag{7.26}$$

This is the measurement update of the error covariance matrix of the Kalman filter equations and is a type of Ricatti equation. The Kalman filter gain can now be calculated by setting the derivative of $dP_k^{+}/dK = 0$, which results in:

$$K = P_k^{-}C^T\left(CP_k^{-}C^T + R\right)^{-1} \tag{7.27}$$

The Kalman filter gain can be substituted in Eq.(7.26), which results in the measurement update of the error covariance matrix found in most textbooks.

$$P_k^+ = (I - KC)P_k^- \tag{7.28}$$

The version of the Ricatti equation in Eq. (7.26) is obtained without substituting the Kalman gain into the Ricatti equation and is called the *Joseph stabilised version* (Lewis, 1986). It guarantees that the state error covariance matrix maintains positive semi-definite and with round-off in computer implementations it should be the preferred version. Thus Eq. (7.28) should be avoided as it cannot be guaranteed to be positive definite.

Linear Mean Square Estimate

The statistic derivation is obtained from the linear mean-squares estimates of the state based on the assumption that the state can be estimated from the present data by a linear dependency:

$$\hat{X}(Y) = AY + b \tag{7.29}$$

It is important to note that this linear dependency should not be thought as a linear model for the observation equation of the Kalman filter. It is the estimate of X given data Y, thus it is a linear approximation of the conditional distribution.

$$\hat{X}(Y) = E\{x|y\} = \int_{-\infty}^{\infty} x \cdot f(x|y)dx = \int_{-\infty}^{\infty} x \frac{f(x,y)}{f(y)}dx = AY + b \tag{7.30}$$

For almost all cases it is troublesome to calculate this conditional estimate, except in the Gaussian case and therefore the linear dependence is mostly used. This also, as the linear dependence assures that only first and second moments are necessary for calculating the estimate.

The coefficients A and b of Eq. (7.29) can be obtained by minimisation of the mean-square error of x, where the estimate $\hat{x}$ is given by Eq. (7.29):

$$J = \overline{(x-\hat{x})^T (x-\hat{x})} = trace\overline{(x-\hat{x})(x-\hat{x})^T} \tag{7.31}$$

The conditions for a minimum, dJ/dA and dJ/b equal to 0, lead to the optimal mean-square estimate:

$$\hat{X}_{LMS} = \overline{X} + P_{XY} P_z^{-1} \left(Y - \overline{Y}\right) \tag{7.32}$$

In Eq. (7.32) no assumption is done about the distributions of X and Y, only the linear dependence of the estimate of E{X|Y} is considered to be linear and is Linear Mean Square Estimate.

Eq. (7.32) has a very appealing appearance and the behaviour of the Kalman filter can be read directly from it. The estimate of X is equal to the mean of X plus a weighting of the covariance between X and Y by the variance of Y, which has a direct appealing understanding of the estimate. If P_{XY} is small than there is a small dependence of X on Y and there will thus be a small gain in the knowledge of X by measuring Y. If the variance of the measurement is very large, then the measurement has a small confidence and X will be adjusted a little by the measurement Y.

The linear mean-square estimate is unbiased, which means that:

$$\overline{\hat{X}}_{LMS} = \overline{\overline{X} + P_{XZ} P_z^{-1} \left(Y - \overline{Y}\right)} = \overline{X} + P_{XY} P_z^{-1} \left(\overline{Y} - \overline{Y}\right) = \overline{X} \tag{7.33}$$

and therefore error covariance associated with the linear mean-square estimate is:

$$P_X = \overline{\left(X - \hat{X}_{LMS}\right)\left(X - \hat{X}_{LMS}\right)^T} = P_X - P_{XY} P_Y^{-1} P_{YX} \tag{7.34}$$

From which it can be seen that the error covariance estimate has the same properties as the mean, if the correlation between X and Y is low, thus P_{XY} low, then the error covariance of X, P_X, is not lowered by a small degree as the measurement Y does not contain that much information about X. But, if there is a dependence of X on Y, then knowledge of the measurements Y lead to a smaller variance of X than when no measurement would be made. If all variables are Gaussian, than filling in the mean and covariance estimates in Eq. (7.32) and (7.34) lead to the same Kalman filter equations mentioned in Eq. (7.27) and Eq. (7.26). When regarding the state space model mentioned in Eq (7.21) and Eq. (7.22), then the covariance of X and Y is given by:

$$P_{XY} = E\left\{\left(X - \bar{X}\right)\left(Y - \bar{Y}\right)^T\right\} = E\left\{\left(X - \bar{X}\right)\left(X - \bar{X}\right)^T C^T\right\} = P_x C^T \tag{7.35}$$

because the measurement noise and v and the state x are assumed to be independent. The variance of the measurement can be written as a function of the variance of x by the measurement equation, which leads to:

$$P_Y = E\left\{\left(Y - \bar{Y}\right)\left(Y - \bar{Y}\right)^T\right\} = E\left\{\left(C\left(X - \bar{X}\right) + v\right)\left(\left(X - \bar{X}\right)^T C^T + v^T\right)\right\} = CP_x C^T + \tag{7.36}$$

Filling in the covariance matrixes of P_{xy} and P_y in the minimum least squares estimate it can be seen right away that the familiar Kalman filter equation is obtained. The same deduction can be done for the update of the error covariance matrix P_x.

An important thing should be noted (Lewis, 1986):

If X and Y are Gaussian variables, then the optimal mean-square estimate is linear and the Kalman filter will be the best linear estimator, since E{X|Y}(mean square estimate) and $\hat{X}_{LMS}$(linear mean square estimate)are the same.

Extended Kalman filter

The extended Kalman filter equations are not derived as they are the same equations as the equations of the standard Kalman filter mentioned in paragraph 0. The difference is that the state space model is non-linear and at every time step the derivatives of the state transition and observation matrix have to be calculated. Thus if the non-linear state space is described by:

$$x_{k+1} = f\left(x_k, u_k\right) + Gw_k \tag{7.37}$$

$$y_k = h(x_k) + v_k \tag{7.38}$$

then the matrixes A and C are given by

$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x_{k+1/k},u=u_k} \qquad C = \left.\frac{\partial h(x)}{\partial x}\right|_{x=x_{k+1/k}} \tag{7.39}$$

These matrixes have to be calculated every sampling instance and mean an extra computational load.

In the non-linear case it cannot be guaranteed anymore that the filter is the optimal linear mean least square estimator. Therefore it is proposed in the next paragraph to modify the process noise covariance Q and make it a function of the prediction error, this might be especially advantageous in case of system identification.

Modification of the Kalman filter

If the Kalman filter is applied as a parameter estimation or as a non-linear identification tool, then it would be preferable that the Kalman adjusts more if its prediction error increases due to modelling error or due to non-linearity. From the Kalman filter equations it can be seen that the error associated with the state estimate X augments without a measurement if Q is non-zero. From the statistical viewpoint this would mean that the uncertainty of the state has augmented. Now it can be seen from the Kalman filter that if P is bigger the measurement gains more confidence in front of the model confidence: P increases, thus K increases and thus a larger correction is made on the basis of the measurement.

Looking at the measurement equation and assuming Gaussian noises it can be easily seen that:

$$y_k - Cx_k = y_k - \overline{y}_k = v_k \tag{7.40}$$

the prediction error is equal to the Gaussian measurement in the linear case as well as the non-linear case. Residual whiteness is mostly used as a measure of the model fit. Taking the covariance expectation the associated error becomes:

$$E\{(y_k - \bar{y}_k)(y_k - \bar{y}_k)^T\} = P_{Yp} = E\{v_k v_k^T\} = R \qquad (7.41)$$

If P_{Yp} is larger than R then we have a state prediction error, from this it was thought to result in a better estimate if the process noise covariance, Q, would be updated in relation to the prediction error. It were used both empirical and statistical derivations to update the matrix Q.

Empirical approaches adopted to have an estimate for the process noise error matrix Q were:

$$Q_i = q_0 \cdot error(y)^2 \qquad (7.42)$$

$$Q_i = \left(\frac{df}{dx_i}\right)^2 \left(P_{Yp} - R\right) \qquad (7.43)$$

This has a more appealing character as the derivative is used in the extended Kalman filter also and in this way it can be used for non-observed states also. Another possible and attractive would be to use the Kalman filter, as it already determines how the state depends on the measurement Y. In the latter case the function of Q becomes:

$$Q = K\left(P_{Yp} - R\right)K^T \qquad (7.44)$$

This approach has to be tested still, and is hoped to give a further improvement.

Recent developments in extended Kalman filtering

Other new and interesting approaches to the extended Kalman filter include the Unscented Kalman filter of Julier and Uhlmann (1997) and the npr-EKF of Nielsen et al. (2001). This technique uses a well chosen set of points to calculate the mean and covariance. These points are the sigma-poins (eigenvectors) calculated from the square root of the error covariance matrix P. These points are transformed by the non-linear state transfer and/or observation function. The sample mean and sample covariance are calculated from the transformed points and equal the posterior mean and covariance error of the posterior conditional distribution (see Figure 46).

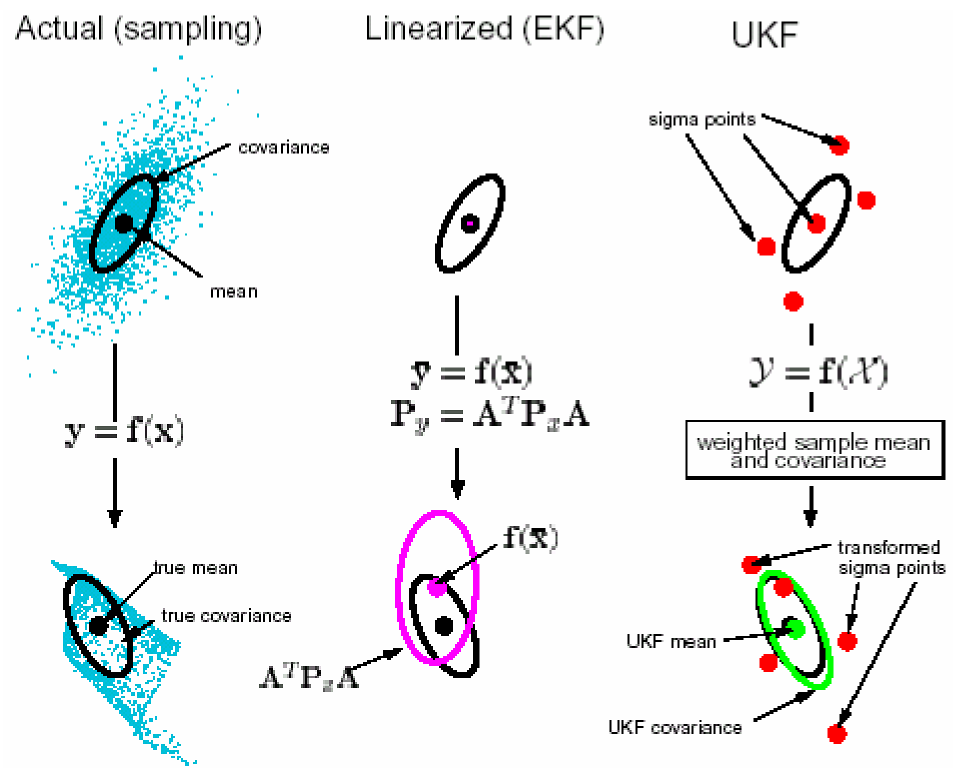


Figure 46: Comparison of the covariance approximation of the unscented Kalman filter and the extended Kalman filter (Merwe and Wan, 2001)

This approach gives an approximation of the second order properties of the posterior conditional distribution function, whereas the EKF gives a correct approximation to the first order only. Thus the unscented Kalman filter will result in a better non-linear estimation, but is not implemented in this work and left as one of the future recommendations.

Other approaches are the use of Monte Carlo methods to generate the probability functions of the anterior and posterior distributions leading to the so called particle filters (Gordon et al. 1993, Doucet et al. 2001). Both unscented Kalman filter and particle filters are based on the principle that it is easier to approximate a probability function than a non-linear function.

Anexo 2 - Otimização com o filtro de Kalman estendido

Resumo – O filtro de Kalman foi utilizado em um ampla gala de problemas para testar a viabilidade dele como algoritmo alternative de otimização aos algoritmos de SQP e algoritmos genéticos. Em problemas sem restrição o filtro de Kalman é mais rápida enquanto com restrições a solução global é encontrada mais vezes.

Este anexo será submetido á revista Computers and Chemical Engineering.

Abstract

In this chapter it is searched for a novel optimisation algorithm with lies in the continuation of the identification methods applied in this work. It was shown (Scheffer and Maciel Filho (2001) that the training of recurrent neural networks can be sped up by the application of a system identification technique, such as the (extended) Kalman filter. It is common to use optimisation techniques for the training of neural networks and therefore it appears that the extended Kalman filter could also be a very effective tool in unconstrained and constrained optimisation. In such a way, the extended Kalman filter could be used to calculate the optimal control action in a model predictive control algorithm or find the most economical set-point in a real-time optimisation algorithm. Probably it is needed more computational resources and thus it has to be analysed if it results in a faster convergence and a better quality of solution.

In the next paragraphs the procedure is pointed out of how to use the Kalman filter in an optimisation scheme, further on some simple examples and the famous banana function of Rosenbrock is shown. The fast solution encountered by the extended Kalman filter algorithm of the Rosenbrock's function shows its high potential for utilisation in more complex constrained optimisation.

It were solved various constrained optimization problems taken from the work of Trvska and Odloak (1998), which they selected because of the occurrence of non-consisted optimisation problems. Therefore these problems seem to be a very suitable set of problems to test the developed Kalman filter optimisation algorithm.

The Kalman filter algorithm shows to be a very stable algorithm without having convergence problems. Some of the problems reach a better global solution, but the solution obtained depends on the initial parameters (P, Q and R) and the way the process noise covariance matrix Q is updated. The convergence speed depends on these parameters also.

Further development of the algorithm should be by application of the "unscented Kalman filter" algorithm, which will avoid the computation of gradienst.

Introduction

A normal minimisation problem can be stated as follows:

$$\begin{aligned} &\min_x f(x) \\ \textit{subjected to}: \quad &g_j(x) \leq 0 \qquad j = 1,\ldots,n_g \\ &h_i(x) = 0 \qquad i = 1,\ldots,n_h \end{aligned} \tag{7.45}$$

where g(x) are the inequality constraints and h(x) the equality constraints with j and i as their indexes.

Eq. (7.45) can be resolved by applying the Langrange multiplier technique. The Langrangian for the problem in Eq. (7.45) is:

$$L(x) = f(x) + \sum_{i=1}^{m} \omega_i \, h_i(x) + \sum_{j=1}^{n} u_j \, g_j(x) \tag{7.46}$$

ω and u are the Langrange multipliers for the equality and inequality constraints respectively. Every constraint gets its own Langrange multiplier.

The minimum, x*, of problem in Eq. (7.45) can thus be resolved by finding the minimum of the langrangian function shown in Eq. (7.46). The necessary conditions and sufficient conditions for this point x* to be a local minimum are summarised in Table 5.

Table 5: Necessary and sufficient conditions for x* to be a local minimum for problem in Eq. (7.45) (Edgar and Himmelblau, 1989)

The *necessary conditions* for x* to be a local minimum of f(x) are:

a) f(x), $h_j(x)$, $g_i(x)$ are all twice differentiable at x*

b) The so-called "second-order constraint qualification" holds (it contains information about the curvature of the constraints that is taken into account at x*; the sufficient conditions for this requirement are that the gradients of the binding constraints ($g_j(x^*)=0$), $\nabla g_j(x^*)$, and the equality constraints, $\nabla h_i(x^*)$, because $h_i(x^*)=0$, are linearly independent.

c) The Lagrange multipliers exist; they do if (b) holds

d) The constraints are satisfied

(1) $h_i(x^*)=0$

(2) $g_j(x^*)\leq 0$

e) The Lagrange multipliers u_j* (at x*) for the inequality constraints are positive (ω_i can be positive or negative)

$$u_j^* \geq 0$$

f) The binding (active) inequality constraints are zero; the inactive equality constraints are <0, and the associated u_j's are 0 at x*

$$u_j^* \, g_j\left(x^*\right) = 0$$

g) The Langrangian function is at a stationary point

$$\nabla\left[L\left(x^*, \omega^*, u^*\right)\right] = 0$$

h) The Hessian matrix of L is *positive semi-definite* for those v's for which $v^T \nabla g_j\left(x^*\right) = 0$, and $v^T \nabla h_i\left(x^*\right) = 0$, that is for all the active constraints

$$v^T \nabla^2 \left[L\left(x^*, \omega^*, u^*\right)\right] v \geq 0$$

The *sufficient conditions* x* to be a local minimum are:

a) The necessary conditions (a), (b) by implication, (c), (d), (e), (f), and (g)

b) Plus a modification of necessary condition (h):

The Hessian matrix of L is *positive definite* for these vectors v such that

$$\left.\begin{array}{l} v^T \nabla g_j\left(x^*\right) = 0 \\ v^T \nabla h_i\left(x^*\right) = 0 \end{array}\right\} \quad \textit{for the binding constraint s}$$

$$v^T \nabla g_j\left(x^*\right) \geq 0 \quad \textit{for the inactive constraint s}$$

$$v^T \nabla^2 \left[L\left(x^*, \omega^*, u^*\right)\right] v > 0$$

These conditions are known as the (Karish)-Kuhn-Tucker conditions, and have to be satisfied for a local or global minimum. Some methods use these conditions as a stopping criterion, while other tries to solve them directly. The well known SQP algorithm and the here developed extended Kalman filter algorithm solve the Kuhn-Tucker conditions directly. Thus, it should be noted that a local or global minimum cannot be distinct by methods using the Kuhn-Tucker conditions and also not by the here developed extended Kalman filter algorithm. Optimisation methods based on genetic engineering seem to have better global convergence properties but these algorithms need a lot of iterations to converge. Other global optimisation methods as the branching method (Ryoo and Sahinidis, 1995) need a large number of iterations to reach the optimum.

It is hoped that the extended Kalman filter algorithm will be fast and able to avoid local minimum, where both can be controlled by adding process noise to the state equations (Puskorius and Feldkamp, 1994, Rivals and Personnaz, 1998; Scheffer and Maciel Filho, 2000 and 2001).

The extended Kalman filter algorithm

The key-point in developing a minimisation algorithm using the Kalman filter is the dynamic system or the state-space equations, which describe the behaviour of the variable to be optimised and the way to observe the state to move it in the direction of the desired state. The starting point for the dynamic system is the Langrangian function of Eq. (7.46), from which two type of state variables can be pointed out, x, the variable to be optimised and ω_i, u_i, the Langrangian multipliers. These variables are optimal if the derivative of the Langrangian and the derivatives of the active constraints are zero, which are mentioned in Table 5 as the necessary conditions for x = x* to be a local minimum. So the states could be adjusted according to the difference from the derivatives to zero. This leads to the following dynamical system, which can be identified by an extended Kalman filter:

$$\underline{x}(n+1) = \begin{bmatrix} x_1(n+1) \\ \vdots \\ x_{nx}(n+1) \\ \omega_1(n+1) \\ \vdots \\ \omega_{n\omega}(n+1) \\ u_1(n+1) \\ \vdots \\ u_{nu}(n+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & & & \cdots & & & & 0 \\ 0 & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & \ddots & & & & \\ \vdots & & & & \ddots & & & & \vdots \\ & & & & \ddots & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & 0 \\ 0 & & & & \cdots & & & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(n) \\ \vdots \\ x_{nx}(n) \\ \omega_1(n) \\ \vdots \\ \omega_{n\omega}(n) \\ u_1(n) \\ \vdots \\ u_{nu}(n) \end{bmatrix} + \begin{bmatrix} w_1(n) \\ \vdots \\ w_{nx}(n) \\ w_1(n) \\ \vdots \\ w_{n\omega}(n) \\ w_1(n) \\ \vdots \\ w_{nu}(n) \end{bmatrix} \qquad (7.47)$$

$$\underline{y}(n)=\begin{bmatrix}\dfrac{\partial L(n)}{\partial x_1(n)}\\ \vdots\\ \dfrac{\partial L(n)}{\partial x_{nx}(n)}\\ \dfrac{\partial L(n)}{\partial \omega_1(n)}\\ \vdots\\ \dfrac{\partial L(n)}{\partial \omega_{n\omega}(n)}\\ \dfrac{\partial L(n)}{\partial u_1(n)}\\ \vdots\\ \dfrac{\partial L(n)}{\partial u_{nu}(n)}\end{bmatrix}+\begin{bmatrix}v_1(n)\\ \vdots\\ v_{nx}(n)\\ v_1(n)\\ \vdots\\ v_{n\omega}(n)\\ v_1(n)\\ \vdots\\ v_{nu}(n)\end{bmatrix} \qquad (7.48)$$

Only the active constraints are taken into account in Eq. (7.47) and Eq. (7.48), and consist of all equality constraints and binding inequality constraints. It is clear from Eq. (7.48) that the observations of the states are the derivatives of the Langrangian to all state variables and need to be zero for a local minimum, which are the necessary condition g and the sufficient condition b when the constraints are binding and u>0. Therefore the "measurement", d, in this case is 0.

The process noise, w_i, and the measurement noise, v_i are normally distributed Gaussian variables. The measurement noise is determined by the type of measurement equipment used. In this case the observation is a mathematical expression, thus the variance of the measurement noise could be chosen as N(0,1). It has to be non-zero to prevent a divide by zero in the matrix inversion in the calculation of the Kalman filter.

The process noise would be zero in case of a parameter, but as mentioned earlier it can be used to influence convergence speed and to avoid local minima. Therefore the variance of the process noise is made an function of the error between the "artificial measurement 0" and the gradient of the Langrangian. Thus the process noise is distributed as the following Gaussian random variable:

$$N(0,\sigma_w(error)) \qquad (7.49)$$

Because of the process noise present, the error covariance matrix has to be updated in the dynamic actualisation of the Kalman filter algorithm as mentioned in chapter 0.

Another way to update the covariance matrix Q is intuitive in that the initial estimate of the parameter to be optimised is bad and thus has a large error. As more adjustments are done the estimate gets better and thus its error and or variance get lower. One of these functions would be:

$$Q(i) = Q_0 * \exp\{-Q_1 * i\} \tag{7.50}$$

Where Q_0 and Q_1 are parameters to be adjusted, i is the current iteration.

The Kalman filter now adjusts every step the state by:

$$\underline{x}^+(k+1) = \underline{x}^-(k) + K(k)(0 - y(k)) \tag{7.51}$$

To calculate the Kalman filter the observation Eq. (7.48) has to be linearised at every sampling moment, which is in this case every calculation step. Linearisation of Eq. (7.48) leads to the second derivative, thus the matrix C of the Kalman filter is in fact the Hessian matrix:

$$C(n) = \begin{bmatrix} \frac{\partial L(n)}{\partial x_1^2(n)} & \frac{\partial L(n)}{\partial x_2(n)\partial x_1(n)} & \cdots & \frac{\partial L(n)}{\partial u_{nu}(n)\partial x_1(n)} \\ \vdots & \vdots & & \vdots \\ \frac{\partial L(n)}{\partial x_1(n)\partial x_{nx}(n)} & \frac{\partial L(n)}{\partial x_2(n)\partial x_{nx}(n)} & \cdots & \frac{\partial L(n)}{\partial u_{nu}(n)\partial x_{nx}(n)} \\ \frac{\partial L(n)}{\partial x_1(n)\partial \omega_1(n)} & \frac{\partial L(n)}{\partial x_2(n)\partial \omega_1(n)} & \cdots & \frac{\partial L(n)}{\partial u_{nu}(n)\partial \omega_1(n)} \\ \vdots & \vdots & & \vdots \\ \frac{\partial L(n)}{\partial x_1(n)\partial \omega_{n\omega}(n)} & \frac{\partial L(n)}{\partial x_2(n)\partial \omega_{n\omega}(n)} & \cdots & \frac{\partial L(n)}{\partial u_{nu}(n)\partial \omega_{n\omega}(n)} \\ \frac{\partial L(n)}{\partial x_1(n)\partial u_1(n)} & \frac{\partial L(n)}{\partial x_2(n)\partial u_1(n)} & \cdots & \frac{\partial L(n)}{\partial u_{nu}(n)\partial u_1(n)} \\ \vdots & \vdots & & \vdots \\ \frac{\partial L(n)}{\partial x_1(n)\partial u_{nu}(n)} & \frac{\partial L(n)}{\partial x_2(n)\partial u_{nu}(n)} & \cdots & \frac{\partial L(n)}{\partial u_{nu}^2(n)} \end{bmatrix} \tag{7.52}$$

Taking in mind that the matrix C is the Hessian matrix, then it appears from Eq. (7.51) that the adjustment of the states is according to the negative gradient of the

Langrangian multiplied by the Hessian matrix. This is just as in Newtons method or as in the SQP algorithm, which is also shown by Bertsekas (1994). The difference is that the here developed Kalman filter algorithm is of a stochastic nature due to the error fed to the states, which can lead to a quadratic convergence and can avoid local minima.

The Kalman gain could be seen as a kind of Newtonian line search method to reach the current local minimum.

Descending direction

As the matrix C is the Hessian matrix it has to be checked if the calculated direction or adjustment results in a smaller value for the objective function. To be a minimisation direction the Hessian matrix has to be positive definite. Eq. (7.52) shows that the matrix C is symmetric for continuous functions and thus the Cholesky decomposition, C=LLT, can be used to factor matrix C into a lower triangular matrix L. In fact the objective function $f(\cdot)$ should be analytic. If the Cholesky decomposition fails, the matrix C is not positive definite, and so the diagonal elements of C should be raised until the Cholesky decomposition succeeds.

Only the elements of the variables x of the original objective function $f(\cdot)$ should be augmented. This because only the variables x are minimised and not the Langrangian multipliers. Only a value of the Langrangian multipliers of the binding constraints has to be found which satisfies the sufficient condition b, so a non-zero value for the equality constraints and a postive value for the binding inequality constraints.

The usage of the positive definiteness can result in small oscillations around the optimum if the second gradient is very high at the minimal solution. This occurs due to the inequality constraints, which are activated or deactivated in these oscillations.

Initialisation of the algorithm

Some optimisation problems can have a linear dependence for some initial point and therefore not reach the optimal solution, for example, problem 3 of the selected problems by Tvrzka de Gouvea and Odloak (1998) Anexo 3. The initialisation of the error covariance matrix, P, can prevent this problem, if the P matrix is not initialised equally for all diagonal elements. For now the P matrix is initialised by:

$$P(n,n) = \begin{bmatrix} 1/n & & \\ & i/n & \\ & & 1 \end{bmatrix} * P_0 \qquad (7.53)$$

Where P_0 is a initialisation parameter.

Active constraint set

It has to be checked if the active inequality constraints fulfil the necessary KKT conditions. This can be done by checking the sign of the Langrange multipliers of the inequality constraints, which have to be positive. If the calculated multiplier is negative, then it has to deleted from the active constraint set. But it was pointed out by Tvrzka de Gouvea and Odloak (1998) that the active inequality constraint set can be very important for the convergence of the SQP algorithm. Tvrzka de Gouvea and Odloak (1998) resolved an extended KKT algorithm using a type of Graham-Schmidt or QR decomposition of the Hessian matrix. In this way it could be identified the linear dependent constraints and project them into the null space.

It was chosen to use every inequality constraint in the active set and thus in the Langrangian function to use all the information about the system at the current value. The non-equal initialisation of the P matrix with the positive definite check avoids this problem and thus it might be more interesting to have the full active set at the current x.

Stopping criteria

The stopping criteria can be based on the absolute or relative difference of the optimal value found or on the gradient difference from zero. In our case the system is stopped on the gradient difference and is:

$$\max_x \nabla L(x) \leq tolerance = 10^{-6} \qquad (7.54)$$

Numerical calculation of the derivatives

The first and second derivatives are calculated numerically to make the algorithm as general as possible. It was implemented a central difference scheme to calculate the Jacobian and Hessian matrix. In the calculation of the derivatives it should be checked if the derivative can be calculated at the x-value. If the initial point is zero, and one of the terms is the root of x, than the pass (x-h) leads to an unreal value.

The first gradient was calculated by:

$$\nabla f(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{7.55}$$

The second gradient was calculated by the following two equations. The diagonal elements are calculated by

$$\nabla f(x) = \frac{\partial^2 f}{\partial x_i^2} = \frac{f(x+2h) - 2f + f(x-2h)}{4h^2} \tag{7.56}$$

while the off-diagonal elements are calculated by:

$$\nabla f(x) = \frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{f(x+h_i, x+h_j) - f(x+h_i, x-h_j) - f(x-h_i, x+h_j) + f(x-h_i, x-h_j}{4h^2} \tag{7.57}$$

Actually both equations are the same.

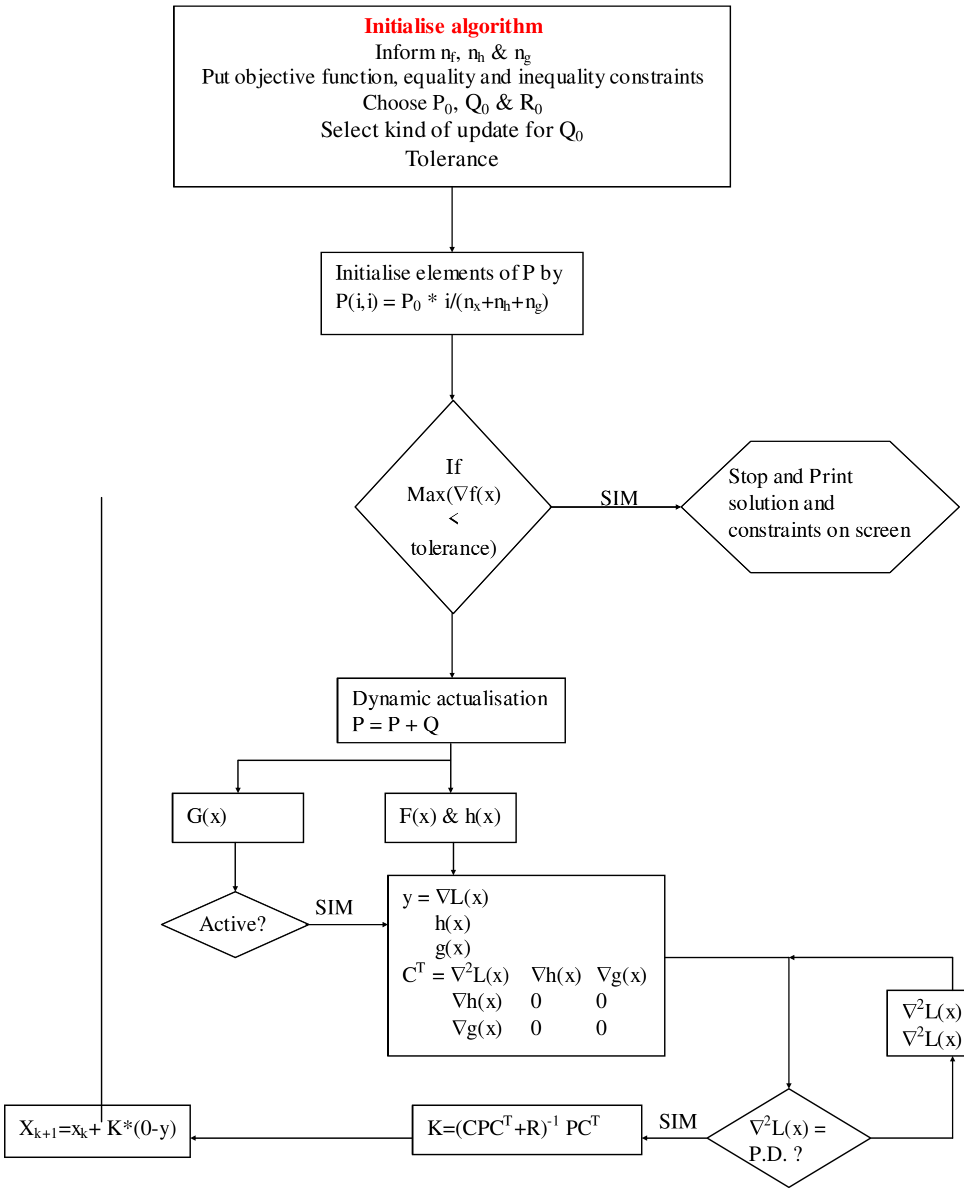
The step size cannot be any number and has to chosen as small as possible, but higher than the machine precision. The step size h should be at least be as high as the root from the machine precision (Numerical Recipes, year) for a backward or forward difference calculation. As it is used the central difference technique, it must be used the 0.25th root. In checking the precision in function of the root taken, it was seen that the square root had a better precision than the 0.25th root. An additional check is done for the current x values according to the lecture syllabus of Trvska de Gouvea (1999). These two considerations lead to the following step size calculation.

$$h = \max(1, x) * (e_{machine})^{0.5} \tag{7.58}$$

So if the maximum x is lower than 1, the step size will be as large as the machine precision.

Block scheme with all the steps of the Kalman filter algorithm

In Figure 47 every step of the extended Kalman filter algorithm in optimization is shown.



Initialise algorithm
Inform n_f, n_h & n_g
Put objective function, equality and inequality constraints
Choose P_0, Q_0 & R_0
Select kind of update for Q_0
Tolerance
Initialise elements of P by
$P(i,i) = P_0 * i/(n_x+n_h+n_g)$
If Max(∇f(x) < tolerance)
SIM
Stop and Print solution and constraints on screen
Dynamic actualisation
P = P + Q
G(x)
F(x) & h(x)
Active?
SIM
$y = \nabla L(x)$
h(x)
g(x)
$C^T = \nabla^2 L(x)\ \ \nabla h(x)\ \ \nabla g(x)$
$\nabla h(x)\ \ 0\ \ 0$
$\nabla g(x)\ \ 0\ \ 0$
$\nabla^2 L(x) = \nabla^2 L(x) + I$
$\nabla^2 L(x)$ = P.D. ?
SIM
$K=(CPC^T+R)^{-1} PC^T$
$X_{k+1}=x_k+ K*(0-y)$

Figure 47: Schematic representation of the extended Kalman filter SQP algorithm

It is emphasised that the gradient is calculated by finite differences. The finite difference algorithm is not optimised yet and can result in a too early stop when the gradient becomes very small. The gradient calculation has a high computational cost for the Kalman filter algorithm as it is needed to calculate the full Hessian matrix. A possibility is to use a Quasi-Newton method to update the Hessian matrix with the gradient vector, but its effect on the stability of the algorithm is not known. Certainly the Unscented Kalman Filter mentioned in paragraph 0 will reduce enormously the computational costs of the algorithm and then result in a fast alternative to SQP algorithms.

Results and Discussion

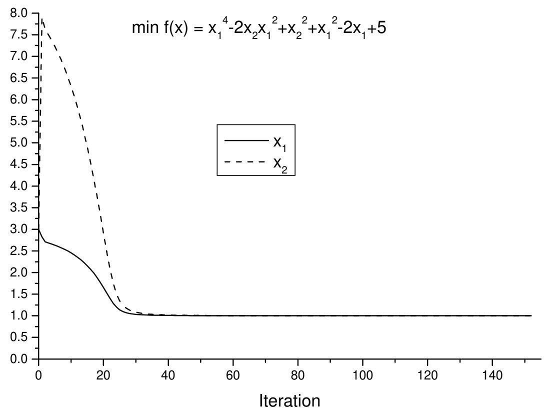
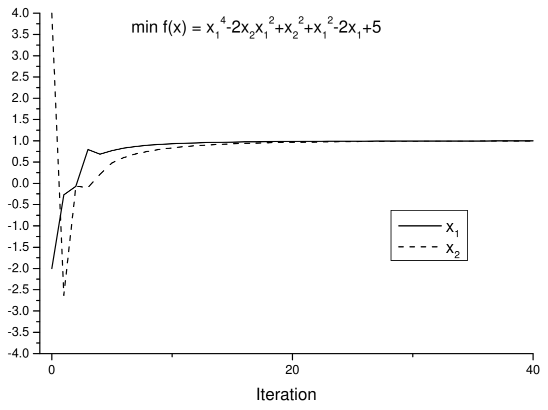
It are shown results from small test problems, which were all taken from Edgar and Himmelblau (1989). Two unconstrained minimisation problems are shown, where the second is known as the Rosenbrock or Banana function. This is because of its characteristic form, which is a canyon in the form of the Banana. This problem is interesting as the gradient in the canyon is very small and leads in case of the steepest gradient method to a slow convergence. The other problem shows the constrained optimisation in two different cases.

Unconstrained minimisation

In Figure 48 and Figure 49 the unconstrained optimisation of the following function are shown:

$$f(x_1, x_2) = x_1^4 - 2x_2x_1^2 + x_2^2 + x_1^2 - 2x_1 + 5 \tag{7.59}$$

The two figures differ in initial conditions, but it can be seen that in both cases x1 and x2 converge fast to the optimal solution. In both cases it appears that the initial converge is rapid but then delays to fulfil the stopping criteria. Thus it can be concluded that the function for the variance of the process noise should be modified to obtain a rapid convergence in initial and end-phase.

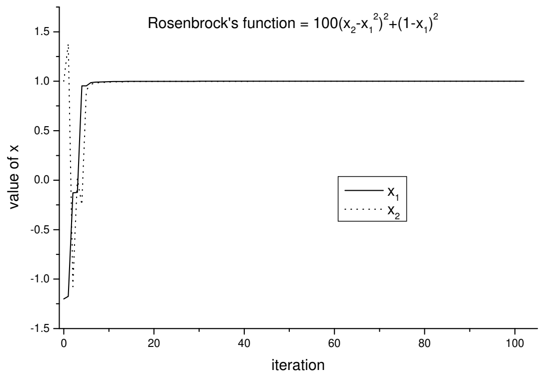
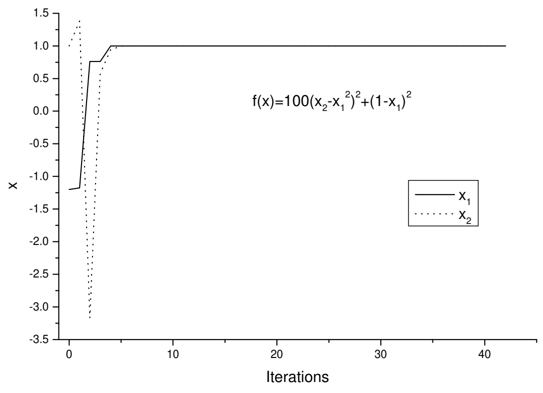
 Figure 48: Unconstrained	 Figure 49: Unconstrained

minimisation with the Kalman filter, q = f(error) = $(d-y)^2$	minimisation with the Kalman filter, q = f(error) = $(d-y)^{0.5}$, other initial conditions

The minimisation of the Rosenbrock function can be seen in Figure 50, Figure 51 and Figure 52, where different functions for the variance of the process noise are used. A quadratic function results in a fast adjustment in the beginning but delays too much to reach the stopping criteria (Figure 50). This is because of squaring the error, which results in even smaller variance for errors below 1, so the error covariance matrix P will not become larger to make more adjustment. From Figure 51 it appears that a square root function results in a faster end-phase training: the stopping criteria is reached in this case within 40 iterations instead of 100. Therefore the two functions are combined in the following way:

$$fN(0,\sigma_w(error)) = \begin{cases} N\left(0,(error)^2\right) & error > 1 \\ N\left(0,(error)^{0,5}\right) & error \leq 1 \end{cases} \tag{7.60}$$

It is hoped that this combination leads to an overall faster convergence in both phases. This can be verified in Figure 52, showing that the combination results in a little bit faster convergence though close to the result of the square root function. Therefore more test are necessary to find the best function for the process noise variance.

 Figure 50: Minimisation of the Rosenbrock's function by the extended Kalman filter, q = $(d-y)^2$	 Figure 51: Minimisation of the Rosenbrock's function by the extended

	Kalman filter, q = (d-y)$^{0.5}$

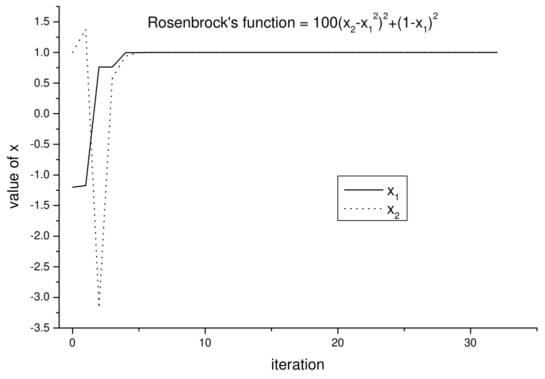


Figure 52: Minimisation of the Rosenbrock's function by the extended Kalman filter, q = (d-y)^2 if (d-y)>1 and q = (d-y)$^{0.5}$ if (d-y)<1

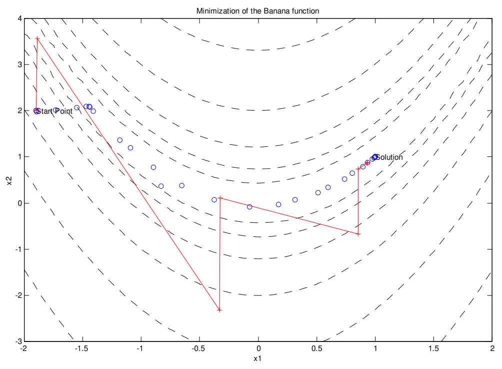


Figure 53: Minimisation of the Rosenbrock's function by the extended Kalman filter (Q = constant, 14 function evaluations) and the Levenberg-Marquardt algorithm (Matlab, 29 function evaluations)

P = 100, q = 10 R = 1e-3

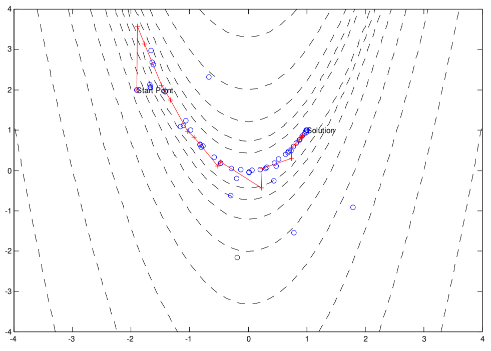


Figure 54: Minimisation of the Rosenbrock's function by the extended Kalman filter (Q = constant, 14 function evaluations) and the BFGS algorithm (Matlab, 54 function evaluations)

P = 100, Q = 10, R = 10

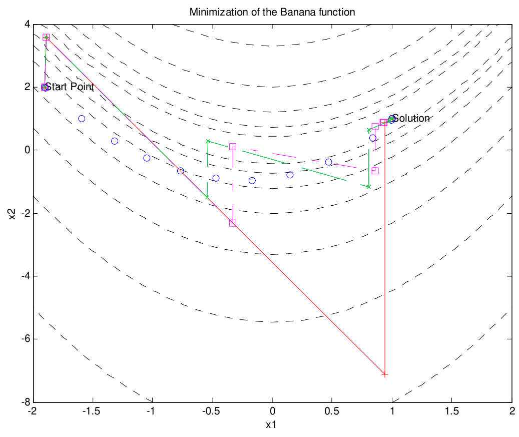


Figure 55: Minimisation of the Rosenbrock's function by the extended Kalman filter comparing analytical versys numerical gradients

Curiously the optimisation with the EKF did not differ that much when the functions were changed. The optimisation path changes if the observation noise is raised, which results in smaller adjustments. The optimisation in Matlab uses analytical gradients, which explains the small difference in solution obtained.

From Figure 55 it can be seen that the calculation of the numerical gradients is still short-coming, especially when the gradient becomes small and the stepsize should be increased to gain accuracy. For the moment the numerical gradient algorithm was not further developed as it is wanted to see the behaviour of the EKF in constrained non-linear optimisation.

Edgar & Himmelblau (1989) show the convergence of various optimisation methods, such as the conjugate gradients method and the Broyden-Flethcher-G-S method. Edgar and Himmelblau mention function evaluations and iterations. For example the conjugate gradient method made 90 function evaluations in the line search method. The Kalman filter algorithm uses positive definite checks for the matrix C to assure a negative direction, which can be seen as function evaluations also.

In Figure 56 and Figure 57 the convergence is shown for the various methods. It can be seen that in the Kalman filter is a very fast minimisation algorithm. In only about 4 iterations it is close to the minimum, but than delays to reach the final stopping criteria. As it is not known what the stopping criteria of Edgar and Himmelblau (1989) was, it is difficult to compare final convergence. Comparing values mentioned in their book shows that the Kalman filter algorithm is faster, but as in the other case a better variance function should be found to fasten convergence in the end-phase.

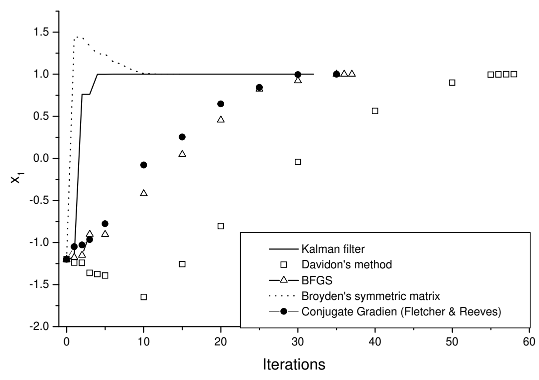


Figure 56: Comparison of the optimisation methods for the Rosebrock function (x1)

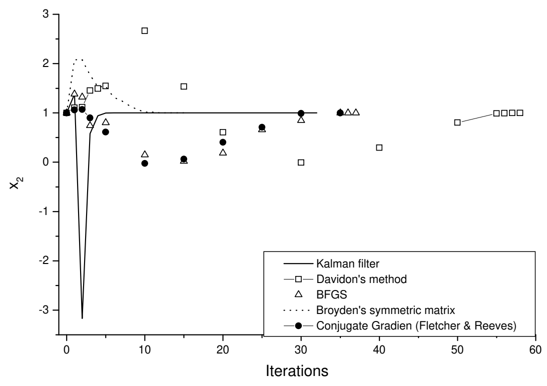


Figure 57: Comparison of the optimisation methods for the Rosebrock function (x2)

It is interesting to see that the path taken by the extended Kalman filter algorithm is totally different than the other algorithms. The initial adjustment is very large due to the high initial value for the P matrix. Then it rapidly settles to the optimal solution.

Constrained optimisation

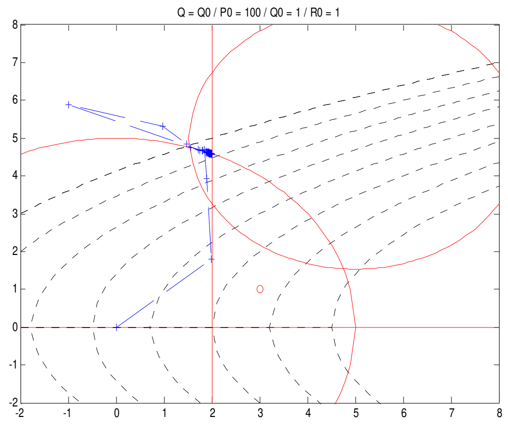
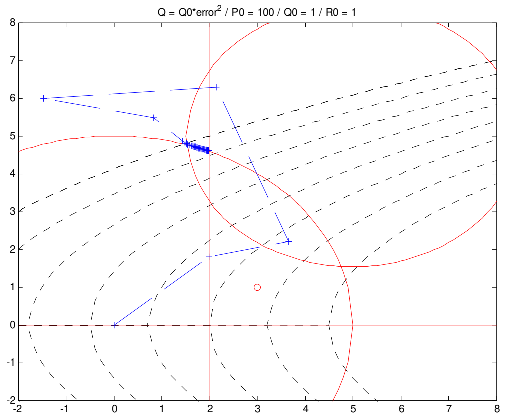
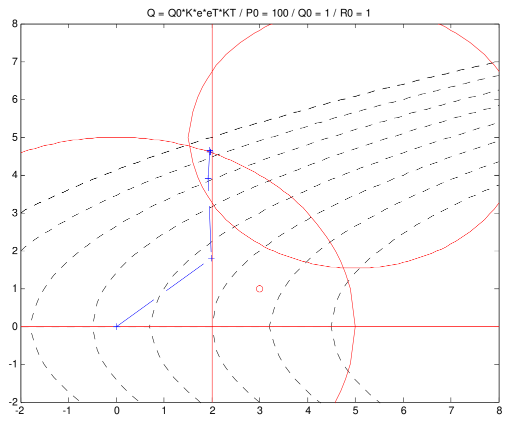
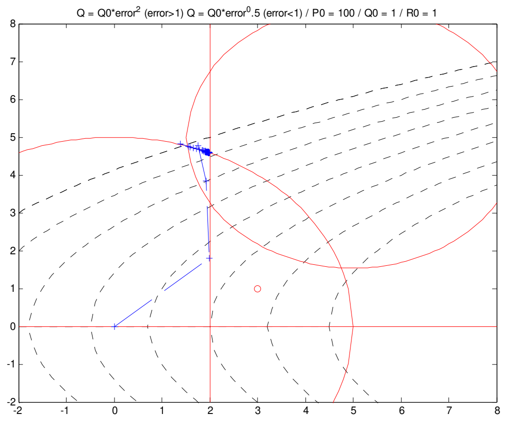
The first constraint minimisation problem showed here was taken of Edgar and Himmelblau (1988) and is described by the following system of equations:

$$\min \quad f(x) = 4x_1 - x_2^2 - 12 \tag{7.61}$$

$$\text{s.t.} \quad h_1(x) = 25 - x_1^2 - x_2^2 = 0 \tag{7.62}$$

$$\begin{aligned} g_1(x) &= x_1^2 + x_2^2 - 10x_1 - 10x_2 + 38 \le 0 \\ g_2(x) &= -(x_1-3)^2 - (x_2-1)^2 \le 0 \\ g_3(x) &= 2 - x_1 \le 0 \\ g_4 &= -x_2 \le 0 \end{aligned} \tag{7.63}$$

Observe in Eq (7.63) that the second inequality constraint is only active in the point (3,1). The results are shown in Figure 58, Figure 59 and Table 6. Observe the differences in convergence for the different functions used for Q and R. It appears that some process noise has to be added for a fast convergence and that making Q a function of the error results in a faster convergence and less oscillation about the optimal solution.

 Figure 58: Q = Q0 (P0 = 100, Q0 = 1 & R0 = 1) (Edgar & Himmelblau, 1989)	 Figure 59: Q = Q0*error^2 (P0 = 100, Q0 = 1 & R0 = 1) (Edgar & Himmelblau, 1989)
 Figure 60: Q = Q0*K*e*e^T*K^T (P0 = 100, Q0 = 1 & R0 = 1) (Edgar & Himmelblau, 1989)	 Figure 61: Q = Q0*error^2 if error>1 and Q = Q_0*error$^{0.5}$ if error<1 (P0 = 100, Q0 = 1 & R0 = 1) (Edgar & Himmelblau, 1989)

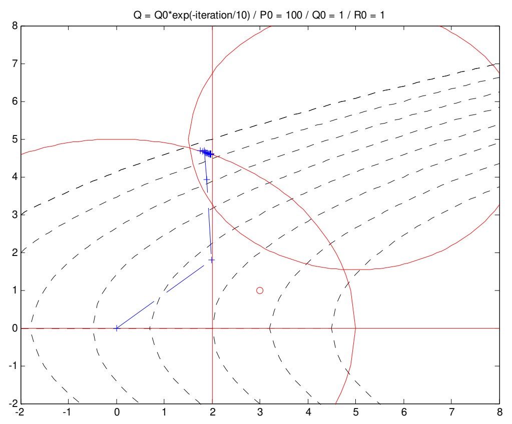


Figure 62: Q = Q0*e$^{(-\text{iteration}/10)}$ (P0 = 100, Q0 = 1 & R0 = 1) (Edgar & Himmelblau, 1989)

The fixed process noise covariance (Figure 58) goes rapidly to the optimal solution also, but as it comes close to the optimal solution it ultra-passes a little bit the optimal solution and the binding inequality constraint becomes non-active. This results in the peak to x = (-1,6), but only happens after about 60 iteration.

It is interesting to see how the solution path and the convergence rate change by making the process noise a function. It seems that the approach of $Q = Q0*K*e*e^T*K^T$ (Figure 60) is the most promising approach as it goes directly to the right solution.

The process covariance noise as a function of the square error, seems to be an alternative approach which might interesting in view of its solution path in global optimization algorithms. As it does not go directly to the closest minimum, but visits all regions around the optimal solution.

Diminishing the process covariance noise by the number of iterations (Figure 62) prevents the oscillation, which was observed for a fixed process noise covariance.

Table 6 : Constrained minimisation with different functions for Q and R applied to the problem of Edgar and Himmelblau (1989)

Q	R	n° iterations	x_1	x_2	ω_1	u_i
Eq. (7.60)	1.0	562	1.9990	4.5830	-1.0000	u_3= 7.9985
Eq. (7.60)	0.001	159	2.0002	4.5825	-1.0000	u_3= 7.9992
1.0	0.001	8	4.7555	1.5445	0.4533	u_1= -0.6477
0.0	0.001	42176*	4.7556	1.5444	0.4462	u_1=-0.6455
1.0	1.0	1107	2.0004	4.5824	-1.0000	u_3=7.9998
1000.0	1.0	499	2.0000	4.5826	-1.0000	u_3=8.0010

* was ended early because of slow convergence

Test problem from Trvska de Gouvea & Odloak (1998)

It were taken various test problems from Trsvka de Gouvea and Odloak (1998), which are known to have convergence problems. These problems include infeasible quadratic sub-problems, singularities and non-convex problems, for which various known SQP-algorithms do not converge (Trvska de Gouvea and Odloak, 1998).

These sample tests are very interesting due to the problems mentioned and therefore form an ambiguous set to submit the Kalman filter algorithm and encounter its performance in terms of its stability and convergence behaviour.

In comparison with normal SQP algorithms, it has to be emphasised that no line-search is conducted with the Kalman filter algorithm. Even though in some cases the Kalman filter can converge more rapidly to the optimal solution. All problems are mentioned in Anexo 3. It was chosen a maximum of 1000 iterations for convenience purposes, sometimes it was run some more iterations to see if the solution was stable or not.

In the next paragraphs a discussion is given of the influence of the tuning parameters P_0, Q_0 and R_0 and the type of update function chosen for the process noise covariance matrix Q.

In Table 7, it are shown only the best solutions obtained by the SQP-EKF algorithm. The complete overview of the results can be conferred Anexo 4. In these tables it can be conferred the influence of the different process covariance noise matrixes and of the intial parameters.

It can be seen that the convergence speed and the solution obtained is very dependent on the initial parameters. Depending on the parameters the system can even converge to an infeasible solution.

Normally the convergence speed increases if the Q is large and the R small, the initial value has some influence on the convergence but seems to have no definite value. Sometimes a small initial value of P leads to a high convergence speed also

Most problems have the largest convergence speed with Q is constant, only very sensitive functions have problems with a large Q and need to have a small value.

In particular it should be mentioned problem 6, which is very difficult due to initial condition chosen. The x's first have to get into the feasible region, but to this was possible only by adding the identity matrix to the matrix CT. The only theoretical basis was to make the CT matrix positive definite, but by accident the identity matrix was added every iteration. To reduce its influence, the identity matrix was devided by the number of iterations done.

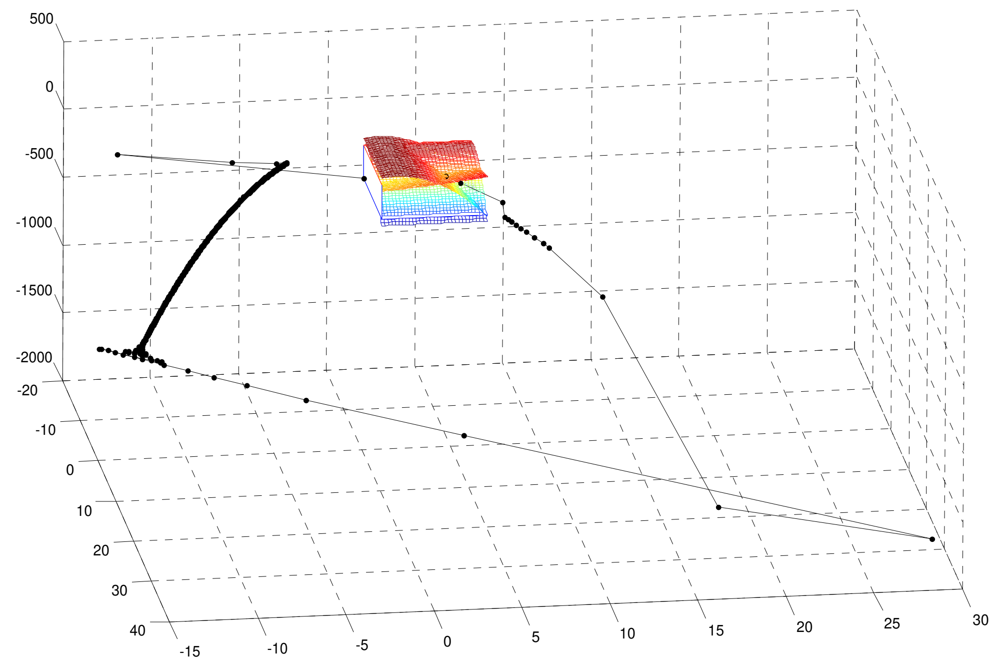


Figure 63: Solution path for problem 6 by adding the matrix added devided by the number of iterations from the point [0, 0, 0, 0] to point [6.29343, 3.82184, 201.15930, 0.00000]

Though the addition of the identity matrix to the CT matrix resolved a very difficult problem, which leads to a non-feasible solution or local minimum for various algorithms, it also resulted in oscillations for some other problems, making it necessary to make it available as a option in the program. It this case it was sufficient to add the identity matrix for the equality and inequality equations to reach the wanted solution.

In Figure 64 it is shown the solution of problem 7, which was solved with the addition of the identity matrix also. Interesting is the capability of crossing the quadrants as the second inequality constraint is not continuous at the origin. In this case it was sufficient also to add the identity matrix for the equality and inequality equations to reach the wanted solution.

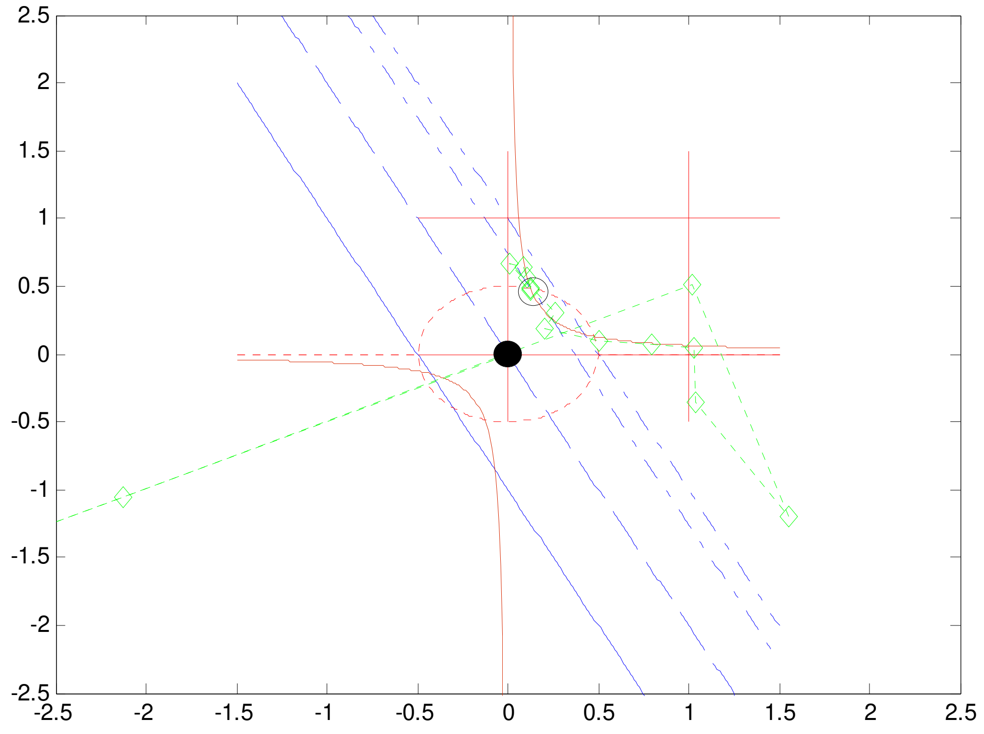


Figure 64: Solution path for problem 7 by adding the matrix added devided by the number of iterations from the point [0, 0, 0] (filled black circle) to point [0.129, 0.483, 0] (open black circle)

In Table 7 of Anexo 4, a full overview is shown of the solutions obtained with the extended Kalman filter constrained optimisation algorithm. The first important thing which can be noted is that the Kalman filter algorithm leads normally to the Global solution. Some problems were not solved due to the combination of initial point and the numerical gradient algorithm calculation implemented. The number of iterations is in the constrained case larger than the SQP algorithm, which makes the calculation time much larger due to the first and second numerical gradient calculation. The iterations are also larger as it is perturbed the Hessian matrix with the identity matrix, but this is necessary to obtain the global solution and can make the algorithm contour constraints.

In some cases it can be seen that the number of iterations can become prohibitive large for more complex problems. Therefore it is thought that this work should be continued in developing a Unscented Kalman filter constrained optimisation algorithm (see paragraph 0), which does not need the calculation of derivatives and use direct function calculations and thus resulting in a tremendous speed-up of the algorithm.

But as can be seen in Figure 65 for problem 4 of the list in Anexo 3, the Kalman filter algorithm normally leads to a rapid adjustment in the beginning and that in a lot of cases the number of iterations is a function of the stopping criteria used.

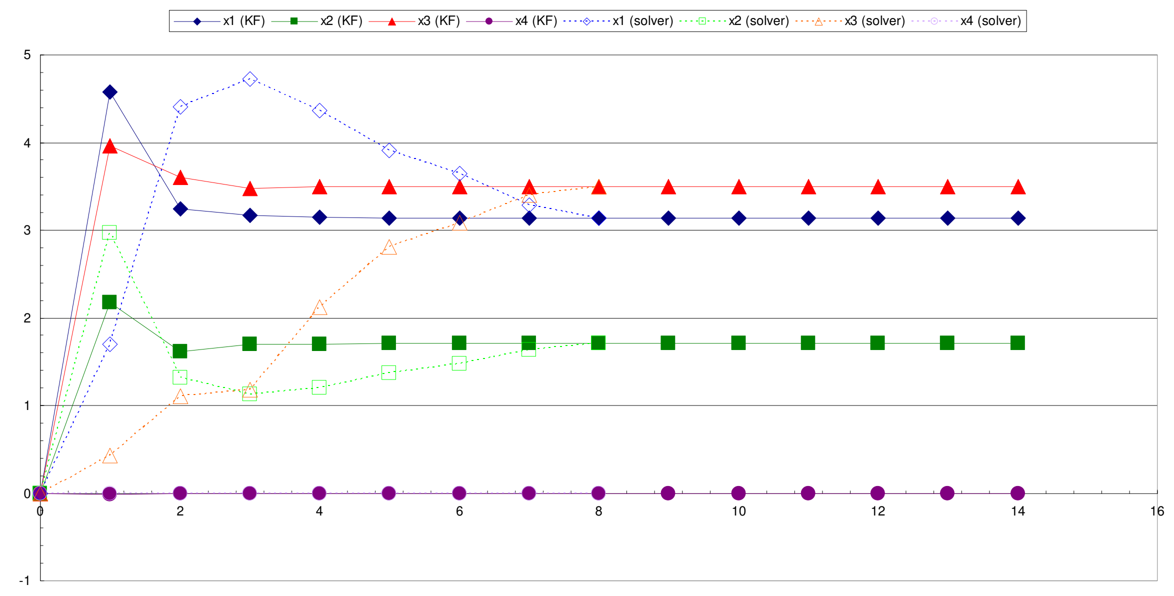


Figure 65: Solution path of the Kalman filter algorithm and Excel Solver (SQP) for problem 4

Constrained optimisation of the phenol oxidation section

It was used the detailed model of the phenol oxidation section developed by Camarasa et al. (2001), which is a validated model with industrial data and propriety of Rhodia Brasil Ltda. The model consists of a hydrodynamic model and a kinetic model of the main reactions occurring in air lift reactions where the oxidation occurs of Cumeno to Cumen Hydro-Peroxide (CHP) mentioned in mentioned in paragraph. The main impurities formed are the DiMethyl Phenyl Carbinol (DMPC), Acetophenone (ACPH) and DiCumylPeroxide (DCP) (Messina et al., 1986). The main point is to minimise the impurities formed in the cumune oxidation subjected to the wanted production, total allowable airflow and the maximum allowed oxygen concentration in the reaction.

Thus the minimisation problem can be described by:

$$\begin{aligned}
f &= \min\left(DMPC + PDC + ACPH\right) \\
h &= CHP_{out,calc} - CHP_{setpoint} = 0 \\
g &= \phi_{m,calc} - \phi_{m,compressor} \le 0 \\
g &= \%O_{2,reactor,calc} - \%O_{2,reactor,\max} \le 0
\end{aligned} \tag{7.64}$$

The problem was resolved with the Bartholew-Biggs algorithm called OPALQP, which is a type of SQP algorithm. But with recent model modifications the algorithm was not optimising anymore.

All data shown in this paragraph was kindly liberated by Rhodia Brasil Ltda. but scaled for proprietary reasons.

To obtain convergence it had to be turned off the addition of the identity matrix to the Hessian or C matrix. Therefore we conclude that the optimisation algorithm has to be tested for every problem if the addition of identity matrix improves the global convergences or not. In the case the phenol optimisation the gradients are very small and the addition of the identity matrix perturbs makes the algorithm go to no impurities and thus also no reaction, as the equality constraint is not obeyed by the summing of the identity matrix!

In Figure 66 and Figure 67 it is shown the convergence of the Kalman filter algorithm and the SQP/OPALQP algorithm. It can be seen that the Kalman filter is faster in the beginning but than slows down its convergence and becomes slower than the OPALQP algorithm. The Kalman filter approaches in one step the feasible region, but then its convergence slows down. This can be a problem of the initial covariance matrixes, but the testing time up to convergence becomes too large. One other reason can be the weighting of the constraints, which is done by the OPALQP algorithm and influences the convergence properties. Including weighing or scaling factors for the constraints in the EKF algorithm might increase its convergence speed also.

One of the weaknesses of using the actual EKF optimisation algorithm, is its computational costs to run the algorithm due to the second gradient calculation of the Hessian or C-matrix of the extended Kalman filter. Thus it should be simplified the Hessian matrix calculation by the BFGS-algorithm or implement the Unscented Kalman filter algorithm, which does not need a gradient calculation.

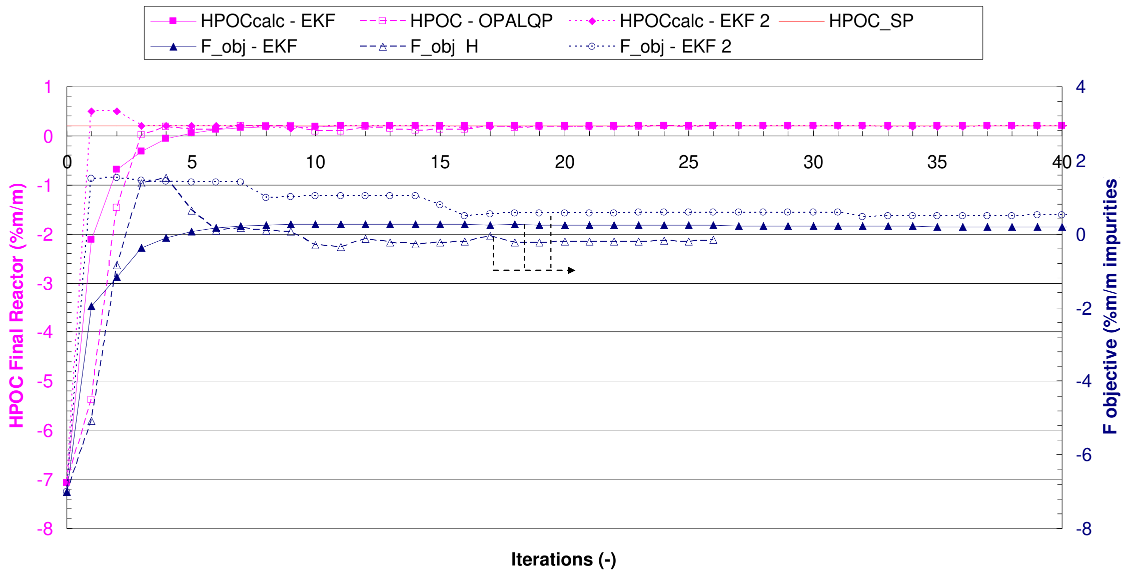


Figure 66: Objective function and equality constraint evolution (production constraint) in case of the Kalman filter algorithm (EKF P_0 = 100, Q_0 = 1, R_0 = 1 and Q_{update} = constant / EKF 2 P_0 = 10, Q_0 = 10, R_0 = 0.01 and Q_{update} = constant) and the SQP/OPALQP algorithm for the industrial optimization

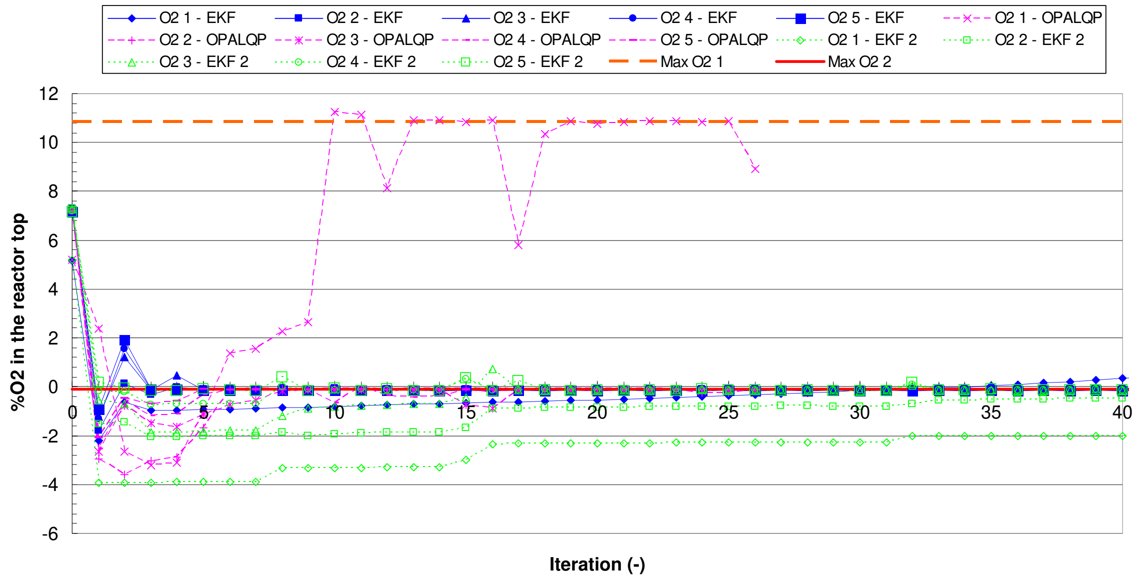


Figure 67: Inequality constraints evolution (oxygen content in top of the reactor) in case of

the Kalman filter algorithm (EKF $P_0 = 100$, $Q_0 = 1$, $R_0 = 1$ and Q_{update} = constant / EKF 2 $P_0 = 10$, $Q_0 = 10$, $R_0 = 0.01$ and Q_{update} = constant) and the SQP/OPALQP algorithm for the industrial optimization

Due to the mentioned high computational load it was implemented the forward difference method for the gradient calculation. In Figure 68 it is shown that the accuracy of the gradient calculation has a large influence on the convergence behaviour. The central difference method does in 1 iteration 8.1 times the step of the forward difference method coming close to the wanted production constrained. The central difference method needs about 3.6 times more function calls but, even if we compare the convergence in function calls then the central difference method does 7.9 times the step of the forward difference method.

If the perturbation step is increased the convergence is improved but remains worse than the central difference gradient calculation. This fact is due to the Hessian or C matrix, whose accuracy is much more sensible to this calculation.

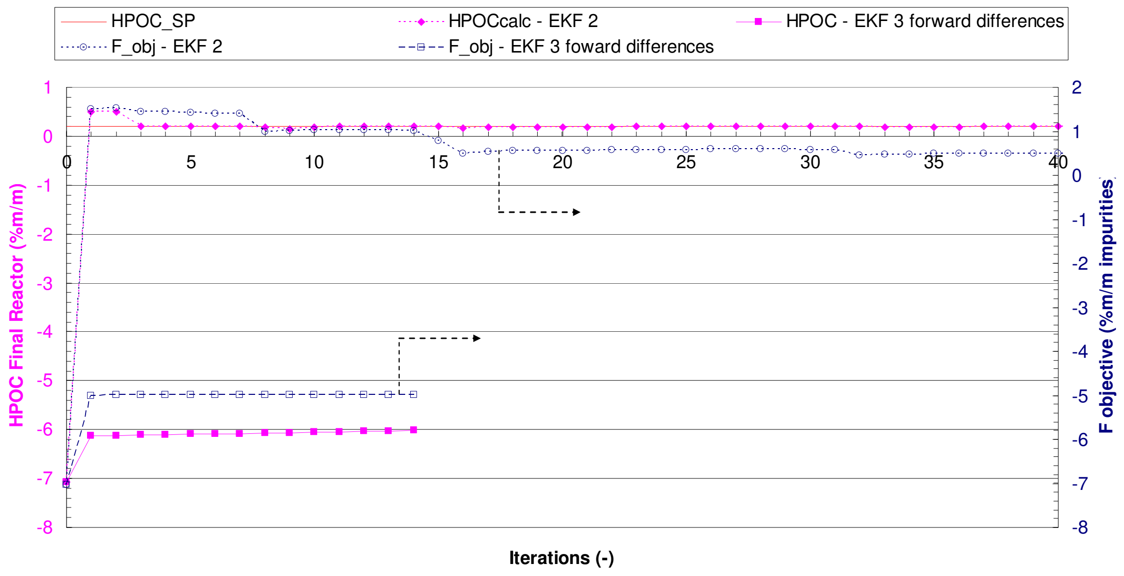


Figure 68: Objective function and equality constraint evolution in case of the Kalman filter algorithm with gradient and Hessian calculation by the central or forward differencing technique

Although there are some problems still with the implementation of the Kalman filter algorithm, its implementation resulted in understanding why the OPALQP algorithm was not working correctly and therefore the OPALQP algorithm could be corrected to work properly as for the moment its solution is still faster than the Kalman filter algorithm.

Conclusions and future work

It was shown that the Kalman filter can be successfully used in unconstrained and constrained optimisation. The unconstrained optimisation of the Rosenbrock function demonstrates that a very fast optimisation can be obtained by manipulating the process noise covariance matrix. The applicability to constrained optimisation was shown in a large scope of different test problems and one real industrial problem. For some of the constrained optimalisation problems it was needed to add the identity matrix to the Hessian matrix to obtain convergence.

In constrained optimisation it was needed more function calls, but by right tuning of the covariance error matrixes it is possible to obtain a better convergence and reach the global minimum, where other algorithms had problems in reaching the global minimum.

The way the process noise covariance matrix Q has to be updated is not clear, constant or a function of the error, but it is essential to get a good convergence to the global minimum.

One of the remaining problems is the need for the calculation of the Hessian matrix which is very time consuming, therefore it is proposed to implement a method like the BFGS to diminish the time for the Hessian matrix calculation.

More preferentially it should be implemented the algorithm of the Unscented Kalman filter which is a gradient free algorithm and therefore could result in a fast optimisation algorithm with the same or even better global convergence properties.

Anexo 3 - Sample problems from Trvska and Odloak (1998)

Problem 1 (Biegler and Cuthrell, 1985):

Min $x_2 + \frac{1}{2}x_3^2$

s.t $1 + x_1 - x_2^2 \leq 0$

$1 - x_1 - x_2^2 \leq 0$

$x_2 \geq \frac{1}{2}$

Problem 2 (Biegler and Cuthrell, 1985)

Min $x_2 + \frac{1}{2}x_3^2$

s.t $-x_2 + 2x_1^2 - x_1^3 \leq 0$

$-x_2 + 2(1 - x_1)^2 - (1 - x_1)^3 \leq 0$

Problem 3 (Edgar and Himmelblau, 1989)

Min $x_1 x_2 + \frac{1}{2}x_3^2$

s.t $-x_1^2 - x_2^2 + 25 = 0$

$-x_1 - x_2 \leq 0$

Problem 4 (Schmid and Biegler, 1994)

Min $1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1 x_2 - x_1 x_3 + \frac{1}{2}x_4^2$

s.t $x_1^2 + x_2^2 + x_3^2 - 25 = 0$

$8x_1 + 14x_2 + 2x_3 = 56$

Problem 5 (Ryoo and Sahindis, 1995)

Min $\quad -x_1 - x_2 + \frac{1}{2}x_3^2$

s.t $\quad x_1 x_2 \le 4$

$\quad 0 \le x \le (6,4)$

Problem 6 (Ryoo and Sahindis, 1995)

Min $\quad x_3 + \frac{1}{2}x_4^2$

s.t $\quad -x_3 + 250 + 30x_1 - 6x_1^2 = 0$

$\quad -x_3 + 300 + 20x_2 - 12x_2^2 = 0$

$\quad -x_3 + 150 + 0.5(x_1 + x_2)^2 = 0$

$\quad 0 \le x \le (9.422, 5.903, 267.42)$

Problem 7 (Ryoo and Sahindis, 1995)

Min $\quad 2x_1 + x_2 + \frac{1}{2}x_3^2$

s.t $\quad -16x_1 x_2 + 1 \le 0$

$\quad -4x_1^2 - 4x_2^2 + 1 \le 0$

$\quad (0,0) \le x \le (1,1)$

Problem 8 (Ryoo and Sahindis, 1995)

Min $\quad -x_1 x_2 + \frac{1}{2}x_3^2$

s.t $\quad 4x_1x_2 + 2x_1 + 2x_2 - 36 \leq 0$

$\quad (0,0) \leq x \leq (1,1)$

Problem 9 (Ryoo and Sahindis, 1995)

Min $\quad -12x_1 - 7x_2 + x_2^2 + \frac{1}{2}x_3^2$

s.t $\quad -2x_1^4 - x_2 + 2 = 0$

$\quad (0,0) \leq x \leq (2,3)$

Problem 10 (Ryoo and Sahindis, 1995)

Min $\quad x_1 + x_2 + \frac{1}{2}x_3^2$

s.t $\quad x_1^2 + x_2^2 - 4 \leq 0$

$\quad 1 - x_1^2 - x_2^2 \leq 0$

$\quad x_1 - x_2 - 1 \leq 0$

$\quad x_2 - x_1 - 1 \leq 0$

$\quad (-2,-2) \leq x \leq (2,2)$

Problem 11 (Ryoo and Sahindis, 1995)

Min $\quad x_1 + x_2 + x_3 + \frac{1}{2}x_6^2$

s.t $\quad \dfrac{100000(x_4 - 100) - 120x_1(300 - x_4)}{a} = 0$

$\quad \dfrac{100000(x_5 - x_4) - 80x_2(400 - x_5)}{a} = 0$

$$\frac{100000(500 - x_5) - 40x_3(600 - 500)}{a} = 0$$

$$(100,100,100,100,100) \le x \le (15834,36250,10000,300,400)$$

a = 1 or a = 100000

Problem 12 (Ryoo and Sahindis, 1995)

Min $x_1^{0.6} + x_2^{0.6} - 6x_1 - 4x_3 + 3x_4 + \frac{1}{2}x_5^2$

s.t $-3x_1 + x_2 - 3x_3 = 0$

$x_1 + 2x_3 - 4 \le 0$

$4 - x_2 - 2x_4 \le 0$

$(0,0,0,0) \le x \le (3,4,2,1)$

Problem 13 (Ryoo and Sahindis, 1995)

Min $\frac{1}{100}\left(35x_1^{0.6} + 35x_2^{0.6}\right) + \frac{1}{2}x_4^2$

s.t $600x_1 - 50x_3 - x_1x_3 + 5000 = 0$

$600x_2 + 50x_3 - 15000 = 0$

$(0,0,100) \le x \le (34,17,300)$

Problem 14 (Ryoo and Sahindis, 1995)

Min $29.4x_1 + 18x_2 + \frac{1}{2}x_3^2$

s.t $6 - x_1 + 0.2458\frac{x_1^2}{x_2} \le 0$

$(0, a) \le x \le (115.8,30)$

a = 0 or a = 0.00001

Problem 15 (Ryoo and Sahindis, 1995)

Min $-x_4 + \frac{1}{2}x_7^2$

s.t $x_1 - 1 + k_1 x_1 x_5 = 0$

$x_2 - x_1 + k_2 x_2 x_6 = 0$

$x_3 + x_1 - 1 + k_3 x_3 x_5 = 0$

$x_4 - x_3 + x_2 - x_1 + k_4 x_4 x_6 = 0$

$x_5^{0.5} + x_6^{0.5} - 4 \le 0$

$(0) \le x \le (1,1,1,1,16,16)$

$k_1 = 0.09755988$

$k_2 = 0.9 * k_1$

$k_3 = 0.0391908$

$k_4 = 0.9 * k_3$

Problem 16 (Psiaki and Park, 1995)

Min $x_2 + \frac{1}{2}x_3^2$

s.t $(x_1 - 1)^2 + x_2^2 + 10000(x_1^2 + x_2^2 - 1)^2 - 0.0625 \le 0$

Problem 17 (Psiaki and Park, 1995)

Min $\frac{1}{2}\{(x_4 - x_2)(x_3 - x_1) + x_8^2\}$

s.t $(x_6 - x_4)(x_3 - x_1) - (x_5 - x_1)(x_2 - x_4) = 0$

$$x_5 - x_7(x_4 - x_2) = 0$$

$$x_6 - x_7(x_3 - x_1) = 0$$

$$-x_5^2 - x_6^2 + 1 \le 0$$

$$x_j + 1 \le 0 \qquad \text{j = 1,2}$$

$$-x_j \le 0 \qquad \text{j = 3,4,5,6,7}$$

Anexo 4 - Tables with the influence of P, Q and R and the type of process noise covariance update on the convergence properties of the SQP-EKF algorithm

Table 7 : Comparison of the results of Tvrska and Gouvea (1998) with the results of the extended Kalman filter algorithm

problem	x0								μ0	μ1	Solution Trvska & Odloak (1998)								Type of	Iter	Solution of Kalman filter algorithm								Type of	Type of algorithm	P	Q	R	Iteration
-	1	2	3	4	5	6	7	8	0	0	1	2	3	4	5	6	7	8	sol		1	2	3	4	5	6	7	8	sol					
1	0	0	0						1	1	0	1	0						Global	10	0	1	0						Global	Q=q0, P = P0/i (i=1-nx)	10	0.001	0.0001	23
1	24	5	0						1	1	0	1	0						Global	9	0	1	0						Global	Q=q0, P = P0/i (i=1-nx)	10	0.001	0.0001	20
2	0	0	0						0	0	0.500	0.375	0						Global	3	0.500	0.375	0						Global	Q=q0, P = P0/i (i=1-nx)	10	0.001	0.0001	21
2	0.8	1	0						0	0	0.500	0.375	0						Global	4	0.500	0.375	0						Global	Q=q0, P = P0/i (i=1-nx)	10	0.001	0.0001	35
2	-0.6	1	0						0	0	-0.618	1	0						Local	5	0.500	0.375	0						Global	Q=q0, P = P0/i (i=1-nx)	1	0.001	0.0001	26
2	1.7	1	0						0	0	1.618	1	0						Local	4	0.500	0.375	0						Global	Q=q0, P = P0/i (i=1-nx)	10	0.001	0.0010	24
3	0	0	0						1	1	-3.536	3.536	0						Global	5	-3.536	3.536	0						Global	Q=q0, P = P0/i (i=1-nx)	100	10	0.01	38
4	0	0	0	0					0	0	4.242	1.847	-1.896	0					Local	21	3.137	1.707	3.499	0					Global	Q=q0, P = P0/i (i=1-nx)	10	0.001	0.0001	20
5	0	0	0						0.9	0.9	2	2	0						Local	6	6	0.667	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	100	0.0001	83
5	3	2	0						0	0	6	0.667	0						Global	6	6	0.667	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	100	0.0001	49
6	9.422	5.903	267.42	0					0	0	6.293	3.822	201.2	0					Global	7	6.293	3.822	201.159	0.000					Global	Q=q0, P = P0/i (i=1-nx)	0.1	1	1	35
6	0	0	0	0					0	0	0	0	150	0					Infeasible	6	6.293	3.822	201.159	0.000					Global	Q=q0, P = P0/i (i=1-nx)	0.1	1	1	857
6	0	0	0	0					0	0	6.293	3.822	201.2	0					Global	4														
6	9.422	5.903	267.42	0					0	0	6.293	3.822	201.2	0					Global	7														
7	0.5	0.5	0						0.2	0.2	0.447	0.224	0						Stationary	6	0.1294	0.483	0.00						Global	Q=q0, P = P0/(nx-i) (i=1-nx	0.001	0.001	0.1	77
7	0.5	0.5	0						0	0	0.483	0.1294	0						Local	8	0.1294	0.483	0.00						Global	Q=q0, P = P0/(nx-i) (i=1-nx	0.001	0.001	0.1	77
7	0.5	0.5	0						0.1	0.1	0.129	0.483	0						Global	7														
7	0.5	0.5	0						0	0	0.129	0.483	0						Global	7														
7	1E-05	0	0						0	0	0.483	0.1294	0						Local	7	0.1294	0.483	0.00						Global	Q=q0, P = P0/(nx-i) (i=1-nx	0.001	0.001	0.1	27
7	1E-05	0	0						1	1	0.129	0.483	0						Global	8														
7	1E-05	0	0						0	0	0.129	0.483	0						Global	8														
8	0	0	0						0	0	0.5	0.5	0						Global	7	0.5	0.5	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	1	1	29
9	0	0	0						0	0	0.718	1.47	0						Global	6	0.7175	1.4698	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	1	1	37
9	1	1.5	0						0	0	0.718	1.47	0						Global	11	0.7175	1.4698	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	1	1	30
10	0	0	0						0	0	1	0	0						Local	8	-1.414	-1.414	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	0.1	0.0001	1000
10	-2	-2	0						0	0	-1.41	-1.414	0						Global	7	-1.414	-1.414	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	0.1	0.0001	1000
10	2	2	0						0	0	1	0	0						Local	7	-1.414	-1.414	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	0.1	0.0001	1000
11	9700	18125	5000	200	25	0			1	1	579.3	1360	5110	182	295.6	0			Global	6	579.3	1 360.0	5 110.0	182.0	295.6	0.0			Global	Q=q0, P = P0/(nx-i) (i=1-nx	10	10	1.00E-06	20339
11	9700	18125	5000	200	25	0			1	1	579.3	1360	5110	182	295.6	0			Global	47														
11	0	0	0	100	100	0			1	1	579.3	1360	5110	182	295.6	0			Global	56	595.0	1 354.8	5 099.6	183.3	296.0	0.0			Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.001	0.1	100000
11	0	0	0	100	100	0			1	1	579.3	1360	5110	182	295.6	0			Global	28														
12	0	0	0	0	0				0	0	0.049	4	1.284	0	0				Stationary	32	due to numerical gradient calc													
12	1.5	2	1	0.5	0				0	0	1.33	4	0	0	0				Global	22	1.33	4.00	0.00	0.00	0.00				Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	10	10	107
12	3	4	2	1	0				0	0	1.33	4	0	0	0				Global	3	1.33	4.00	0.00	0.00	0.00				Global	Q=q0, P = P0/(nx-i) (i=1-nx	100	10	10	59
13	0	0	100	0					0	0	33.33	0	300	0					Global	3	0	16	100	0										
14	57.9	15	0						0	0	8.17	7.56	0						Global	12	8.17	7.56	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.01	0.001	27
14	0	1E-05	0						0	0	8.17	7.56	0						Global	14	8.17	7.56	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.01	0.001	26
15	0	0	0	0	0	0	0		0	0	0.963	0.499	0.03681	0.3858	0.3981	11.35	0		Local	21	due to numerical gradient calc													
15	0	0	0	0	0	0	0		1	1	0.772	0.517	0.2042	0.389	3.037	5.096	0	-0.3746	Global	28														
15	0.5	0.5	0.5	0.5	8	8	0		0	0	1	0.393	0	0.3881	16	0	0		Local	13	0.791	0.511	0.189	0.392	2.716	5.532	0.0	-0.392	Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.001	0.1	7907
15	0.5	0.5	0.5	0.5	8	8	0		1	1	0.391	0.391	0.3746	0.3746	16	0	0		Local	24	0.791	0.511	0.189	0.392	2.716	5.532	0.0	-0.392	Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.001	0.1	7907
16	0	0.99	0						0	0	not converge								-	17	0.9687	-0.2481	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.01	0.001	430
16	0	0.99	0						1	1	0.969	-0.248	0						Global	300	0.9687	-0.2481	0						Global	Q=q0, P = P0/(nx-i) (i=1-nx	1	0.01	0.001	360
16	0	0.99	0						1	1	0.969	-0.248	0						Global	218														
17	-2	-3	6	6	6	6	6	0	1	1	not converge								-	61	ht that there is some error in the mentioned problem													
17	-2	-3	6	6	6	6	6	0	0	0	-1	-1	2.41	2.41	0.707	0.707	0.207	0	Global	300														
17	-2	-3	6	6	6	6	6	0	0	1	-1	-1	2.41	2.41	0.707	0.707	0.207	0	Global	78														
17	-1	-1	0	0	0	0	0	0	0	0	-1	-1	2.41	2.41	0.707	0.707	0.207	0	Global	21														

Anexo 5 – Developed programs in this thesis

In this work there were developed various Fortran programs, which are available on a CD with the orientador of this thesis at the laboratory LOPCA/UNICAMP.

1. External Recurrent Neural Network training program with the training algorithms, backpropagation, backpropagation with momentum, the Multiple Extended Kalman filter algorithm and the Global Extended Kalman filter algorithm. The backpropagation algorithms can be used in a batch or sequential training mode. Teacher-forcing can be used in the recurrent neural network mode.
2. STABLE (Extended) Kalman filter algorithm for state adjustment or parameter adjustment. The algorithm is stable as the *Joseph stabilised* version is used
3. Constrained non-linear optimization algorithm based on the Extended Kalman filter algorithm
4. Fractional Gaussian Noise modeling and Hurst's parameter estimation by the Maximum Likelihood method. It can be used simulate fractional landscapes also.
5. A model to simulate the penicillin process

One of the programs is not available as it is propriety of Rhodia Brasil Ltda..

6. A model to simulate the phenol oxidation section