Speech Enhancement with Factor-Graphs

Diplomarbeit

durchgeführt von

Wolfgang Irnberger

am

Institut für Signal- und Informationsverarbeitung
der Eidgenössischen Technischen Hochschule Zürich

Institut für Elektronische Musik und Akustik
der Universität für Musik und Darstellende Kunst Graz

Vorstand: o.Univ.-Prof. Mag. DI Dr. Robert Höldrich

Betreuer und Begutachter: o.Univ.-Prof. Mag. DI Dr. Robert Höldrich
Externer Betreuer: DI Sascha Korl (ETH Zürich)

Graz, im März 2003

Kurzfassung

In dieser Arbeit wird versucht, Sprachsignale mittels Faktor-Graphen zu bearbeiten und damit die Sprachverständlichkeit gestörter Signale zu verbessern. Dazu wurde das Linear Predictive Coding (LPC)-Modell zur synthesischen Spracherzeugung herangenommen und in die Faktor-Graphen Darstellung übergeführt. Dieses erste Modell wird in dieser Arbeit als Ausgangpunkt für weitere Verbesserungen und Abänderungen herangezogen.
Kapitel 2 gibt eine kurze Einleitung in die Theorie der Faktor-Graphen. Es wird die Darstellung, der Umgang und die Berechnung von Faktor-Graphen erläutert. In Kapitel 3 wird das vorhandene LPC-Modell in der Faktor-Graphen Darstellung erklärt. Linear Predictive Coding kommt aus der digitalen Sprachverarbeitung und wird zur Sprachsynthese und auch Analyse verwendet. Kapitel 4 befasst sich mit der Erweiterung des Graphen um ein Störmodell und Modifizierungen bei den Filtergewichten um diese besser nachbilden zu können. Diese Veränderungen sollen das gesamte Modell verbessern. Um reale Sprachsignale in Echtzeit verarbeiten zu können, wird in Kapitel 5 der Aufbau des Faktor-Graphen dynamisiert. Damit soll der Rechenaufwand minimiert werden. Diese Dynamisierung und die Auswirkungen der in Kapitel 4 eingeführten Veränderungen werden in Kapitel 6 getestet.
Die verwendete Programmier- und Simulationsumgebung ist C++.

Abstract

In this diploma thesis I tried out to handle speech signals with the help of factor-graphs. The aim is to improve the audibility of speech in noisy backgrounds. For this the linear predictive coding (LPC) - model was already transformed into a factor-graph. This factor-graph was now changed in this diploma thesis to fit better to speech signals.
Chapter 2 is a small introduction to factor-graphs, how to construct and compute them. In Chapter 3 the factor-graph of the linear predictive coding - model is introduced. The LPC is used to analyze and synthesize speech. Chapter 4 deals with the modifications of the factor-graph. A noise was modelled and the filter coefficients were modified to enhance the whole model. To calculate signals with the system in real-time a dynamic construction of the factor-graph was developed, which is described in Chapter 5. In Chapter 6 the modifications are tested and documented.
The programming and simulation was done in C++.

Dank

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meinem Studium und bei dieser Arbeit unterstützt haben.

Bei Sascha Korl am Institut für Signal- und Informationsverarbeitung (ISI) an der ETH Zürich für sein Know How, das er mir bei inhaltlichen Problemen immer zukommen ließ.

Weiters bei Prof. Robert Höldrich vom Institut für Elektronische Musik und Akustik (IEM) an der Kunstuniversiät Graz für seinen Zuspruch zum Auslandsaufenthalt in der Schweiz.

Ganz besonderer Dank gebührt meinen Eltern **Hermann** und **Ernestine Irnberger**, die durch Ihren Fleiß und Ihre Arbeit mir mein Studium erst ermöglicht haben und mich immer, nicht nur finanziell, unterstützt haben.

Danke auch meiner Freundin, die mich in den letzten beiden Jahren meines Studiums immer tatkräftig unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Eine noch sehr junge Art und Weise der Modellbildung und Algorithmenbehandlung sind Faktor-Graphen, welche in ganz unterschiedlichen Fachgebieten, wie z.B. der Signalverarbeitung, der Codierung/Dekodierung, bei Markov-Feldern in der Physik, Anwendung finden. In dieser Arbeit werden nun Einsatzmöglichkeiten von Faktor-Graphen in der Sprachsignalverarbeitung behandelt. Auch hier wurde ein bekanntes Modell zur Sprachanalyse und -synthese, nämlich das Linear Predictive Coding (LPC)-Modell, in Faktor-Graphen übergeführt.
Wenn sich zwei Personen unterhalten kann das durch folgendes Modell dargestellt werden (Abbildung 1.1):
Die Signalquelle ist der Sprecher (rechts), der die Audiosignale aussendet. Seine Laute breiten sich im Raum in zwei akustischen Kanälen aus und gelangen an das linke und rechte Ohr des Zuhörers (links), der dann die Senke darstellt. Beeinflusst wird dabei das Audiosignal durch die Raumeigenschaften (Faltung mit der Raumimpulsantwort) des Ortes an dem sich Zuhöhrer und Sprecher befinden. Eine weitere Manipulation erfolgt durch andere Audiosignale aus dem Umfeld, welche in unser System als additive Störquellen einfließen.
Wenn man nun als Zuhörer eine hörbehinderte Person annimmt, so ist unser Ziel mit Hilfe obigen Modells Audiosignale so zu manipulieren, dass sich für die Person eine möglichst hohe Sprachverständlichkeit ergibt. Der Faktor-Graph berechnet die Signale $\tilde{d}_r[.]$ und $\tilde{d}_l[.]$ bzw. des Sprechersignals $\tilde{u}[.]$. Diese Schätzung soll dann eine möglichst gute Nachbildung der Wirklichkeit sein.
Die Nachbildung der Modelle soll mittels Faktor-Graphen geschehen. Dazu wurde in [4] das Sprachmodell bereits als FFG entwickelt und prinzipiell getestet. In dieser Arbeit wird nun versucht, dieses Modell zu verbessern und an die Anforderungen zur Verarbeitung realer

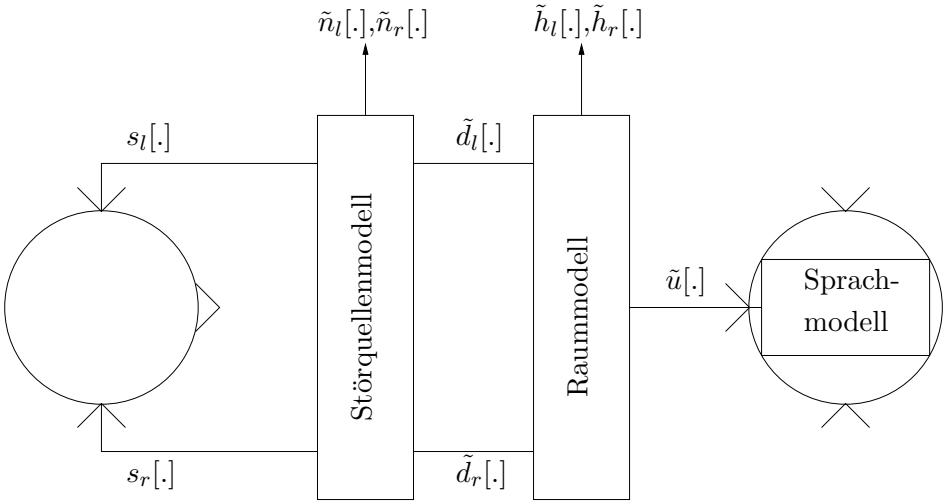


Abbildung 1.1: Sprachsituation als Modell, bestehend aus Störquellenmodell, Raummodell und Sprachmodell dargestellt. Die Größen $\tilde{n}_{l,r}[.]$, $\tilde{h}_{l,r}[.]$, $\tilde{d}_{l,r}[.]$ und $\tilde{u}[.]$ werden durch das Modell geschätzt.

Sprachsignale anzupassen. Dazu muss die Möglichkeit geschaffen werden, mit dem Faktor-Graphen in Echtzeit Signale zu berechnen. Wir versuchen dies durch einen dynamischen Auf- und Abbau eines Faktor-Graphen Modells zu erreichen. Weiters soll die Filterung im Sprachmodell besser modelliert werden und auch die Einbindung eines Störquellenmodells untersucht werden.

Kapitel 2

Faktor-Graphen

In diesem Kapitel wird ein Überblick über die Grundlagen von Faktor-Graphen gegeben. Die genauen Herleitungen und Details sind in [1]und [2] zu finden.

2.1 Einführung

Ein *Forney Faktor-Graph (FFG)* ist ein Diagramm, welches die Faktorisierung einer Funktion darstellt. Allgemein geschrieben:

$$f(x_1, x_2, \ldots, x_n) = \prod_{i \in I} f_i(X_i) \tag{2.1}$$

wobei X_i ein Subset $\{x_1, x_2, \ldots x_n\}$ und I ein Set endlicher Indizes $\{1, 2, \ldots n\}$ ist.
Die globale Funktion f wird in einzelne lokale Funktionen f_i faktorisiert, welche nun nur mehr von einem Subset von $x_1, x_2, \ldots, x_n$ abhängen. So kann zum Beispiel die Funktion $f(a, b, x, y)$ zum Produkt einzelner Funktionen $f_1(a)f_2(b)f_3(a, b, x)f_4(x, y)$ faktorisiert werden.

$$f(a, b, x, y) = f_1(a)f_2(b)f_3(a, b, x)f_4(x, y) \tag{2.2}$$

Den entsprechenden Faktor-Graph dazu zeigt Abbildung 2.1. Im FFG stellt dann ein Knoten eine lokale Funktion dar, die Variablen werden durch Kanten bzw. Halbkanten ersetzt und eine Kante (Halbkante) ist genau dann mit einem Knoten verbunden, wenn die Variable ein Argument der Funktion ist.
Ein Faktor-Graph kann auch als verallgemeinertes Blockschaltbild einer Funktion angesehen werden.

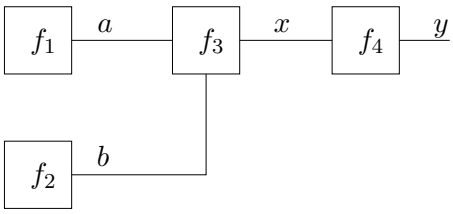


Abbildung 2.1: FFG der faktorisierbaren Funktion

2.2 Berechnungen mit einem FFG

In unserem System rechnen wir mit Wahrscheinlichkeitsverteilungen, die sich mittels der FFG-Darstellung sehr gut berechnen lassen. Dazu betrachten wir die Knoten als Prozessoren und die Kanten als Kanäle, in denen Nachrichten in beide Richtungen weitergeleitet werden. Die Berechnung der einzelnen Randdichten im FFG erfolgt dann mit dem

Summe-Produkt Algorithmus:

$$\mu_{f_A \to x_i}(x_i) = \sum_{\sim\{x_i\}} \left(f_A(x_1, \ldots, x_N) \prod_{n \in \{1,\ldots,N\}\setminus\{i\}} \mu_{x_n \to f_A}(x_n) \right) \tag{2.3}$$

Die Nachricht, welche über die mit der Variablen x_i assoziierten Kante vom Knoten f_A zum Knoten f_B gesendet wird, ist das Produkt der lokalen Funktion von f_A mit allen an f_A anliegenden Nachrichten (außer von f_B herkommend) und einer nachfolgenden Summation über alle Variablen außer x_i. N ist babei die Anzahl der Argumente der Funktion f_A.

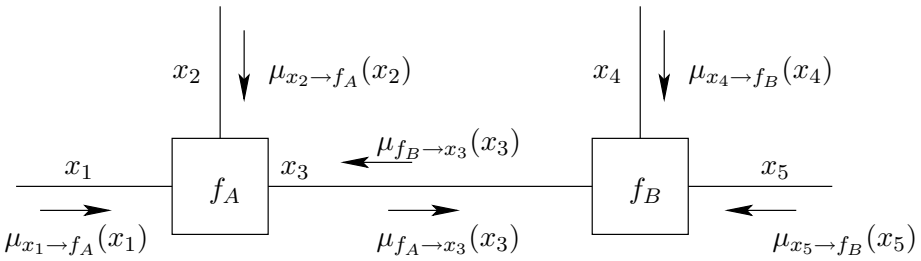


Abbildung 2.2: Nachrichten des Summe-Produkt Algorithmus

Wenn alle Nachrichten berechnet sind, ist der Summe-Produkt Algorithmus beendet und die Randdichte $g_i(x_i)$ zwischen zwei Knoten berechnet sich dann aus dem Produkt der beiden

Nachrichten, die an einer Kante anliegen.

$$g_i(x_i) = \mu_{f_A \to x_i}(x_i)\mu_{f_B \to x_i}(x_i) \tag{2.4}$$

2.3 Scheduling eines FFG

Die Randdichten $g_i(x_i)$ lassen sich in einem Durchgang berechnen, wenn der Graph keine Schleifen besitzt. Dazu muss aber auch eine logische Reihenfolge der Berechnungsschritte vorgegeben werden, damit eine Nachricht an einem Port erst berechnet wird, wenn alle notwendigen Eingangsnachrichten vorhanden sind. Der Summe-Produkt Algorithmus wird am FFG *'straight forward'* angewendet.
Kommen im FFG Schleifen vor, bedeutet dies, dass es Knoten gibt, an denen zum Berechnungszeitpunkt eine benötigte Nachricht noch nicht berechnet ist. Es wird daher notwendig beim Aufbau des Graphen alle Kanäle mit einer Einheitsnachricht zu initialisieren, um einen solchen Knoten trotzdem berechnen zu können. Dies bedeutet aber auch, dass dieser Knoten in einem weitern Durchlauf, einer Iteration, des Summe-Produkt Algorithmus nochmals berechnet werden muss. Die Nachrichten im FFG können damit nicht mehr in einem Durchlauf *'straight forward'* berechnet werden. Je nach Reihenfolge der Anwendung der *'Update-Rules'* aus Abschnitt 3.4 können oft viele Iterationen notwendig sein, bis die Variablen im Graphen konvergieren oder es passiert, dass sie nicht gegen einen fixen Wert gehen. Es soll versucht werden die Anzahl der notwendigen Iteration mit Hilfe eines guten Schedules möglichst klein gehalten werden. Man gibt dem Summe-Produkt Algorithmus also in einer Liste vor, wann welche Ports der Knoten berechnet werden sollen. Das Scheduling bei der Verwendung von Faktor-Graphen ist noch ein sehr unerforschtes Gebiet, es gibt noch keine Standardlösungswege, die allgemein gültig sind.

2.4 Typen von Nachrichten

Für den FFG des Sprachmodells wurden in [4] drei verschiedene Nachrichtentypen entwickelt:

Gauß'sche Nachricht (Kalman Nachricht)[1]:

Die Wahrscheinlichkeitsdichte der Nachricht ist eine Normalverteilung und die Variablen im FFG werden als Gauß'sche Zufallsvariablen modelliert. Charakterisierend für die Normalverteilung wird der Mittelwert m_x und die Varianz V_x zur Berechnung

[1] Bei der Kalman Filterung sind die Variablen durch gauß'sche Zufallsvariablen modelliert.

verwendet.

$$\mathcal{N}(m_x, V_x, x) = \frac{1}{\sqrt{2\pi V_x}} e^{-\frac{(x-m_x)^2}{2V_x}} \tag{2.5}$$

Diskrete Nachricht :

Die Nachricht besteht aus einer diskreten Wahrscheinlichkeitsverteilung

Gain Nachricht :

Die Nachricht am Gainknoten (Tabelle 3.2 (5)) hat die Form

$$\mu_g(g) = \frac{1}{|g|} exp\left(-\frac{(\frac{m_x}{m_y} - \frac{1}{g})^2}{2\left(\frac{V_x}{m_y^2}\right)}\right) \tag{2.6}$$

$$:= \frac{1}{|g|^{k_g}} exp\left(-\frac{(a_g - \frac{1}{g})^2}{2b_g}\right) \tag{2.7}$$

und die Nachricht wird in diesem Fall durch

$$k_g = 1, \qquad a_g = \frac{m_x}{m_y}, \qquad b_g = \frac{V_x}{m_y^2} \tag{2.8}$$

charakterisiert.

2.5 FFG als generatives Modell

Man kann mit Hilfe des Faktor-Graphen Signale selbst generieren. Eigentlich sind Verbindungen im FFG ungerichtet, die Randdichte an einer Kante errechnet sich, wie in Abschnitt 2.2 bereits gezeigt, aus dem Produkt der Nachrichten in beide Richtungen. Beim Erstellen eines Signals werden konkrete Samples im Graphen in Vorwärtsrichtung ermittelt, wobei dazu in den Apriori-Knoten bestimmte Randbedingungen, wie in unserem Fall die Filtergewichte und der Verstärkungsfaktor, vorgegeben werden. So können wir uns ein synthetisches Ausgangssignal $y[n]$ (Abbildung 3.2) zu Simulationszwecken selbst generieren. Es werden also keine Randdichteverteilungen berechnet, sondern ein mögliches Signal, welches dieser Verteilung entspricht. Zur eigentlichen Berechnung mit dem Summe-Produkt Algorithmus werden die Randbedingungen, wie Filtergewichte und Verstärkungsfaktor auf unbekannt gesetzt und das vorher simulierte Signal wird als Eingangssequenz verwendet. Damit lässt sich einfach überprüfen, ob im FFG die Wahrscheinlichkeitsverteilungen, wie die Filtergewichte, richtig berechnet werden.

Kapitel 3

Faktor-Graph des LPC-Modells

Kapitel 3 gibt einen Überblick über den Aufbau des FFG-Modells, welches in [4] erarbeitet wurde.

3.1 Einführung

Der FFG in Abbildung 3.2 ist aus dem LPC-Sprachproduktionsmodell (Abbildung 3.1) entstanden. Die genaue Herleitung der einzelnen Knoten sind in [4] nachzulesen. Der Faktor-Graph, wie er dann zur Berechnung verwendet wird, setzt sich aus einzelnen Slices zusammen, welche die zeitliche Abfolge wiedergeben. Damit am Beginn und Ende keine offenen Zweige (Halbkanten) entstehen, werden sogenannte 'Apriori- Knoten' angehängt, welche eine schon vorgegebene Nachricht vom richtigen Typ besitzen. In der Vertikalen entspricht jeder Slice des FFG dem örtlichen Ablauf eines Samples durch das LPC-Modell. Ein Slice

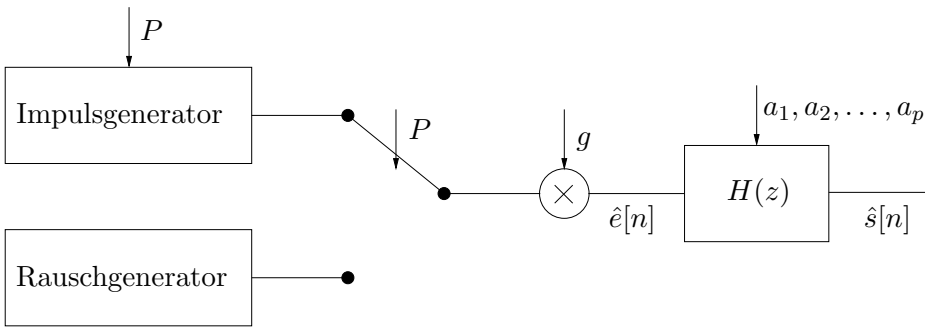


Abbildung 3.1: LPC-Sprachmodell: P, g, und $a_1, a_2, \ldots, a_p$ sind die benötigten Größen zum Steuern der Anregungsfunktion und des Sprachsynthesefilters $H(z)$.

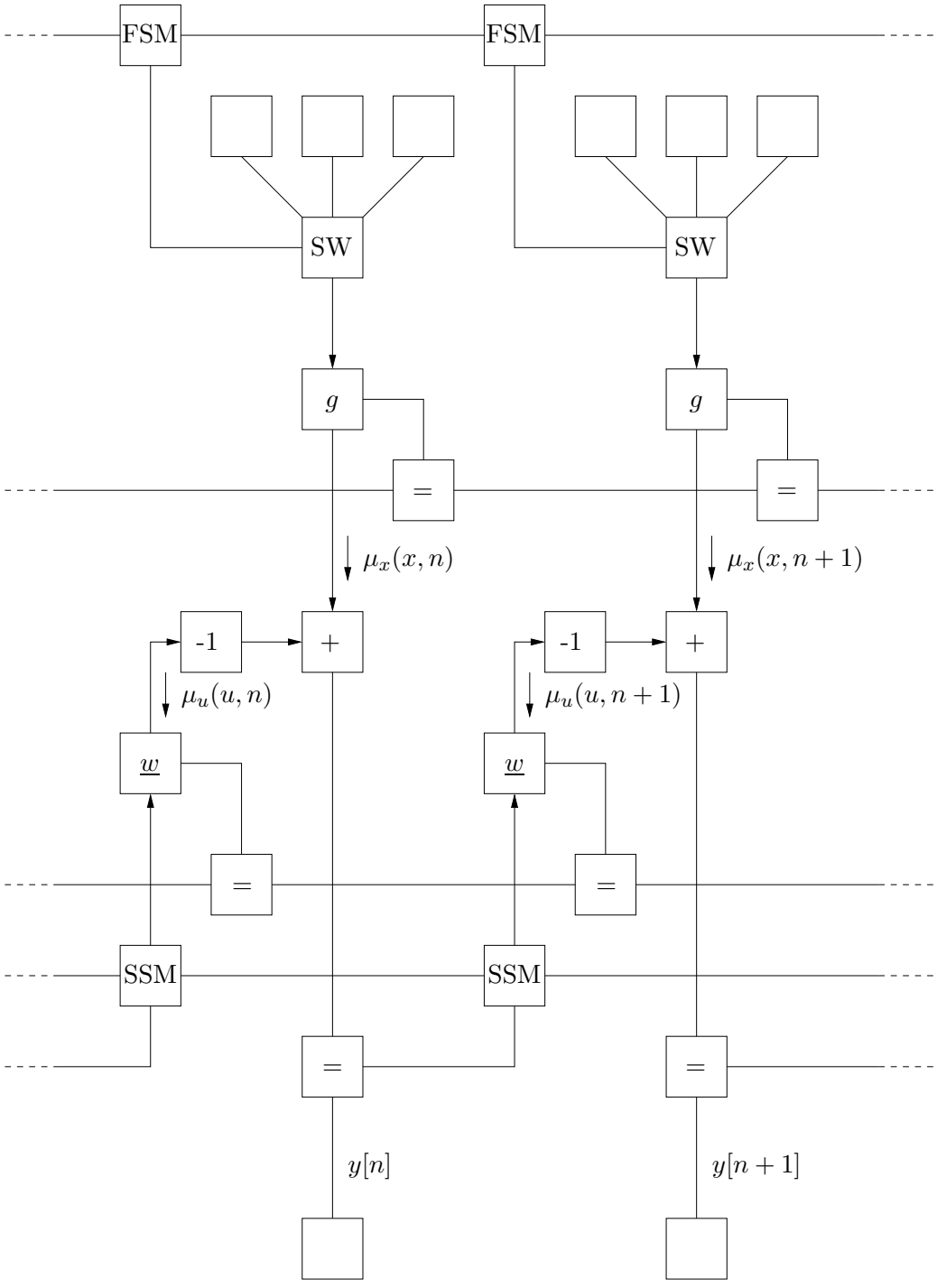


Abbildung 3.2: FFG des LPC-Sprachmodells aus [4]

setzt sich also aus Anregungs- und Vokaltraktmodell zusammen. Das Anregungsmodell wird durch die Finite State Machine (FSM), dem Schalter und dem Gain-Knoten gebildet. Durch die Verbindung der einzelnen Gain-Knoten jedes Slices ergibt sich schon die erste Schleife im FFG. Der zweite Teil spiegelt die Filterung des erzeugten Anregungssignals wider und setzt sich aus einem Kalman-Filter und einem State Space Model zusammen. Durch das Abgleichen der Filtergewichte entsteht eine weitere Schleife.

3.2 Anregungsmodell

Das Anregungsmodell liefert uns das Anregungssignal $\mu_x(x, n)$ (Abbildung 3.2) welches entweder ein Impulszug mit der Periode P oder ein weißes Rauschen ist. Die Steuerung des Schalters, der zwischen stimmhaft (*'voiced high'* und *'voiced low'*) und stimmlos (*'unvoiced'*) wechselt, übernimmt die FSM. Der genaue Aufbau findet sich in [4]. Die FSM beinhaltet eine Matrix $\mathbf{P}_{s_k|s_{k+1}}$ (3.1) mit den jeweiligen Zustandsübergangswahrscheinlichkeiten. Die Periodendauer τ gibt die Frequenz wieder, P_{vv} ist die Wahrscheinlichkeit, dass der Schalter *'voiced'* bleibt, P_{uu} ist die Wahrscheinlichkeit für *'unvoiced'*.

$$\mathbf{P}_{s_k|s_{k+1}} = \begin{pmatrix} 0 & P_{vv} & 0 & \dots & 0 & 1-P_{vv} \\ \vdots & \ddots & P_{vv} & \ddots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & \dots & 0 & P_{vv} & \vdots \\ P_{vv} & 0 & \dots & \dots & 0 & 1-P_{vv} \\ \frac{1-P_{uu}}{\tau} & \dots & \dots & \dots & \frac{1-P_{uu}}{\tau} & P_{uu} \end{pmatrix} \tag{3.1}$$

Mit P_{vv} in der ersten Nebendiagonale wird die Frequenz der FSM konstant gehalten, da im *'voiced'*-Fall alle τ Zustände für *'voiced low'* durchlaufen werden.
Mit dem Gainknoten wird die Lautstärke abgeglichen, die nach der Modellbildung in Abbildung 3.2 über die Slices konstant gehalten wird, da die Knoten durch ein *'equal gate'* verbunden sind.

3.3 Vokaltraktmodell

Die Filterung erfolgt durch ein stabiles Allpol-Filter der Form

$$H(z) = \frac{1}{\sum_{i=0}^{N} w_i z^{-i}}, \qquad w_0 = 1 \tag{3.2}$$

und der entsprechenden Differenzengleichung für den Zeitbereich

$$y[n] = -\sum_{i=1}^{N} w_i y[n-i] + x[n]. \tag{3.3}$$

Am Eingang des Filters liegt ein Vektor mit den letzten N Eingangssamples an. Für den nächsten Slice $N+1$ wird nur der neue Wert in diesen Vektor geschoben. Diese lineare Abhängigkeit des Vektors zwischen den beiden Zeitschlitzen lässt sich mit einem linearen Zustandsraummodell (SSM[1]) sehr einfach nachbilden.

3.4 Update Rules

Tabelle 3.1 und 3.2 zeigen die Rechenregeln für die Kalman Nachricht an den wichtigsten Knotentypen. Die mathematischen Herleitungen sind in [1] und [4] zu finden. Wie in Tabelle 3.1 zu sehen ist, wird bei den Update-Gleichungen der Ein-/Ausgangsnachrichten die Kovarianzmatrix **V** sowie deren Inverse, die Kostenmatrix **W**, verwendet. Wird zum Beispiel im Skalierungsknoten (Tabelle 3.1 (3)) die Nachricht, die vom Eingang x weggeht, berechnet, kommt man nicht um eine Matrixinversion herum. Diese bringt jedoch Schwierigkeiten mit sich.

3.5 Schwachpunkte des Modells

Im Summe-Produkt Algorithmus zur Berechnung der Randdichten im Graphen wurde bis jetzt keine Rückrechnung im Filter- und SSM-Zweig berücksichtigt. Genau in diesem Fall wird die Berechnung mit der Kostenmatrix **W** benötigt. Damit lässt sich die Inversion von Matrizen nicht mehr verhindern.

3.5.1 Inverse bzw. Pseudoinverse einer Matrix

Im Allgemeinen gehen wir von positiv-definiten Matrizen **V** und **W** aus. Absolute Sicherheit bei einem Wert wird dann durch $\mathbf{V} = 0$ ausgedrückt, keine Information durch $\mathbf{W} = 0$. Das Rechnen mit *Null* und beim Invertieren folglich $'NAN'$ [2] bzw. $'INF'$ [3] führt zu Problemen, sodass wir für große Varianz ($\triangleq$ keine Information) $\mathbf{V} = 10^6$ bzw. $\mathbf{W} = 10^{-6}$ verwenden.
Wenn die Matrix quadratisch und singulär ist, existiert keine Inverse. Zur Berechnung kann dann die Pseudo-Inverse (*'Moore-Penrose Generalized Inverse'*) herangezogen werden.

[1] SSM: Akronym für **S**tate **S**pace **M**odel
[2] NAN: C-Notation für **N**ot **A** **N**umber
[3] INF: C-Notation für **Inf**inity

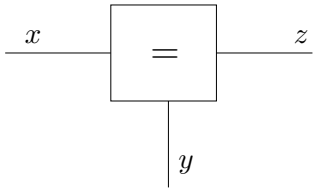
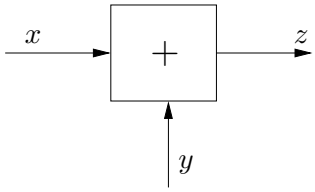
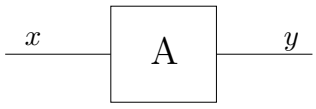
1	Gleichheitsknoten  $\delta(x-y)\delta(x-z)$	$\begin{aligned} m_z &= (W_x+W_y)^{\#}(W_x m_x + W_y m_y) \\ W_z &= W_x + W_y \\ V_z &= V_x(V_x+V_y)^{\#}V_y \\ m_x &= (W_z+W_y)^{\#}(W_z m_z + W_y m_y) \\ W_x &= W_z + W_y \\ V_x &= V_z(V_z+V_y)^{\#}V_y \\ m_y &= (W_x+W_z)^{\#}(W_x m_x + W_z m_z) \\ W_y &= W_x + W_z \\ V_y &= V_x(V_x+V_z)^{\#}V_z \end{aligned}$
2	Additionsknoten  $\delta(z-x-y=0)$	$\begin{aligned} m_z &= m_x + m_y \\ W_z &= W_x(W_x+W_y)^{\#}W_y \\ V_z &= V_x + V_y \\ m_x &= m_z - m_y \\ W_x &= W_z(W_z+W_y)^{\#}W_y \\ V_x &= V_z + V_y \\ m_y &= m_z - m_x \\ W_y &= W_x(W_x+W_y)^{\#}W_y \\ V_y &= V_x + V_z \end{aligned}$
3	Skalierungsknoten  $\delta(y-Ax)$	$\begin{aligned} m_y &= Am_x \\ V_y &= AV_xA^H \\ m_x &= (A^HW_yA)^{\#}A^HW_ym_y \\ W_x &= A^HW_yA \end{aligned}$

Tabelle 3.1: Update Rules der Nachrichten, für Mittelwert m und Kovarianzmatrix $\mathbf{V}$ bzw. $\mathbf{W}$, an den verschiedenen Knoten

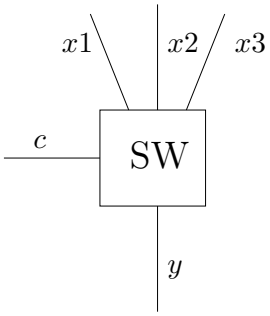
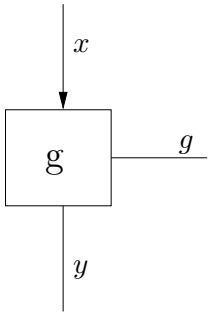
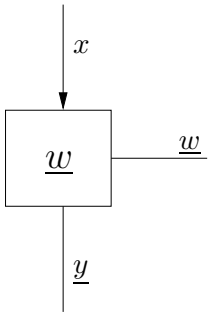
4	Schalter  $\delta(s-s_0)\delta(y-x_0)+$ $\delta(s-s_1)\delta(y-x_1)+$ $\delta(s-s_2)\delta(y-x_2)$	$\begin{aligned} m_y &= \mu_s(s_0)m_{x_0}+\mu_s(s_1)m_{x_1}+\mu_s(s_2)m_{x_2} \\ V_y &= \mu_s(s_0)\big(V_{x_0}+(m_y-m_{x_0})^2\big)+ \\ &\quad \mu_s(s_1)\big(V_{x_1}+(m_y-m_{x_1})^2\big)+ \\ &\quad \mu_s(s_2)\big(V_{x_2}+(m_y-m_{x_2})^2\big) \\ m_{c_i} &= m_{x_i}-m_y \\ V_{c_i} &= V_{x_i}+V_y \end{aligned}$
5	Verstärkungsknoten  $\delta(y-gx)$	$\begin{aligned} m_x &= \frac{m_y}{g} \\ V_x &= \frac{V_y}{g^2} \\ m_y &= m_x g \\ V_y &= V_x g^2 \\ a_g &= \frac{m_x}{m_y} \\ b_g &= \frac{V_x}{m_y^2} \\ k &= 1 \end{aligned}$
6	Filterknoten  $\delta(x-\underline{w}^t\underline{y})$	$\begin{aligned} m_x &= m_y^t m_w \\ V_x &= m_w^t V_y m_w \\ m_y &= \frac{m_x}{m_w^t m_w} m_w \\ W_y &= m_w m_w^t W_x \\ m_w &= \frac{m_x}{m_y^t m_y} m_y \\ V_w &= m_y m_y^t W_x \end{aligned}$

Tabelle 3.2: Update Rules

Moore-Penrose Generalized Inverse :

Die Matrix **A** sei quadratisch und singulär, dann ist

$$\mathbf{B} = pinv(\mathbf{A}). \tag{3.4}$$

Weiters gilt:

$$\mathbf{ABA} = \mathbf{A} \tag{3.5}$$

$$\mathbf{BAB} = \mathbf{B} \tag{3.6}$$

$$\mathbf{AB} \; ist \; hermitesch \tag{3.7}$$

$$\mathbf{BA} \; ist \; hermitesch \tag{3.8}$$

Zur Berechnung der Pseudoinversen wird die *'Singular Value Decomposition'* verwendet. Eine beliebige quadratische Matrix **A** kann durch

$$\mathbf{A} = \mathbf{USV}^H \tag{3.9}$$

dargestellt werden, wobei **U** und **V** unitär sind und **S** eine reale Diagonalmatrix mit nur positiven Einträgen ist.

$$\mathbf{S} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_l \end{pmatrix} \qquad \sigma_k > 0, k = 1 \ldots l.$$

Mit Hilfe der *Singular Value Decomposition* berechnet sich die Pseudo-Inverse der Matrix **A** nun:

$$\mathbf{A}^{\#} = \mathbf{VS}^{\#}\mathbf{U}^H \tag{3.10}$$

mit

$$\mathbf{S}^{\#} = \begin{pmatrix} \sigma_1^{-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_l^{-1} \end{pmatrix}.$$

Bei der *Singular Value Decomposition* werden die *Singular Values* in der Diagonale von **S** Null gesetzt, wenn sie unter der Toleranzgrenze liegen.

$$tol = d * norm(\mathbf{S}) * \epsilon \tag{3.11}$$

d	...	Dimension der zu invertierenden Matrix
$\mathbf{S}$	...	Matrix **S** der SVD
$\epsilon = 2.2204 \cdot 10^{-16}$	...	Abstand von 1.0 zur nächst größten *floating-point* Nummer

Während der Berechnung des FFG kann der Wert $'0'$ in der Diagonale einer Matrix immer noch auftreten. Da man *'INF'* und *'NAN'* bei einer weiteren Inversion vermeiden will, wird in der Hauptdiagonale der Varianz-Matrix ein genügend kleiner Wert ϵ addiert. Bei empirischen Versuchen hat sich herausgestellt, dass ϵ nicht zu klein sein darf , da sonst die *Singular Values* wieder unter die Toleranzgrenze fallen.

3.5.2 Modellbildung

Modellierung von Störsignalen

Da in dem besprochenen bestehenden Modell noch keine Nachbildung eines Störsignals eingebunden ist, können auch keine verrauschten Signale gereinigt werden.

Nachbildung der Filterkoeffizienten

In dem Faktor-Graphen aus Abbildung 3.2 sieht man, dass die Filtergewichte durch die Gleichheitsknoten über die Zeit konstant gehalten werden. Diese Annahme ist eine Vereinfachung, welche zur Nachbildung von realen Sprachsignalen nicht mehr gültig ist, da sich bei der Sprache in der Realität die Filterkoeffizienten ständig ändern können. Daher wird im folgenden Kapitel versucht, diese Bindung der Filterkoeffizienten zu lösen und eine bessere Modellierung zu finden, damit sich die Filtergewichte über die Zeit kontinuierlich verändern können.

Dynamisierung für Echtzeit Berechnungen

Bis jetzt konnten nur kurze, einfache, stationäre Signalen mit dem Faktorgraphen berechnet werden und so die prinzipielle Funktionalität des Modells untersucht werden. Um ein Audiosignal zu berechnen steigt der Rechenaufwand enorm, da über eine große Anzahl von Slices gerechnet werden muss. Um ein Sample zu berechnen sind jedoch nicht alle vorhergegangenen Werte von Interesse, es kann daher immer ein Ausschnitt des Faktor-Graphen für den Summe-Produkt Algorithmus herangezogen werden, der sich dann über die Zeitachse verschiebt. In Kapitel 5 wird der FFG sich dynamisch verändern um so den Rechenaufwand zu minimieren.

Kapitel 4

Erweiterung des LPC Faktor-Graphen

Der bis jetzt besprochene Faktor-Graph stellt ein Grundgerüst dar, welches noch in viele Richtungen erweitert und ausgebaut werden kann. Bis jetzt sind einige Vereinfachungen im Graphen vorgenommen worden, mit denen man Abweichungen von der Realität in Kauf nimmt. So ist auch die Filterstruktur mit all den Problemen bei der Rückrechnung noch nicht am Ende aller Möglichkeiten.
In diesem Kapitel wollen wir ein ganz allgemeines Störquellenmodell einführen, den Gain-Knoten noch genauer betrachten und versuchen, die Filterkoeffizienten besser zu modellieren.

4.1 Störquellenmodell

Um Herauszufinden welchen Einfluss ein Störsignal auf die Berechnungen im FFG hat und ob eine Störung eliminiert werden kann, wird als Störgeräusch ein einfaches weißes Rauschen mit Mittelwert $m_n = 0$ und Varianz $V_n = P_n$ angenommen. Nachgebildet wird dies im Faktor-Graphen, wie in Abbildung 4.1 zu sehen ist, mit einem Apriori-Knoten mit

$$\mu_n(n) = \mathcal{N}(m_n, V_n, n)^1 \tag{4.1}$$

und einem Additionsknoten, der an den Eingang eines jeden Slices des FFG geschaltet wird. Durch das Hinzufügen von weißem Gauß'schen Rauschen wird also im Summationsknoten

[1] Es gibt keine standardisierte Notation für die Gauß'sche Nachricht. Diese Schreibweise wurde von [4] übernommen. Die hier verwendete Notation bedeutet $\mu_n(n)$ ist gaußverteilt mit Mittelwert m_n und Varianz V_n, n gibt die laufende Variable an.

die Kalman-Nachricht Richtung Gleichheitsknoten zu

$$\mu_d(d) = \mathcal{N}(m_y - m_n, V_y + V_n, d) \tag{4.2}$$

Verfolgt man nun den Weg der Eingangsnachricht durch den FFG so passiert Folgendes:

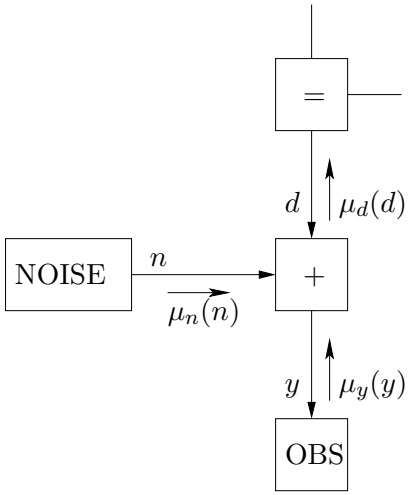


Abbildung 4.1: FFG des Störsignalmodells

Die Eingangsnachricht hat die Varianz $V = 10^{-6}$, ist also ein Minimum. Im Additionsknoten wird diese um die Varianz des AWGN [2] erhöht. Im Gleichheitsknoten am Eingang wird die vergrößerte Varianz weitergereicht zum Filteradditionsknoten, an dem noch die Varianz vom Filter her addiert wird. Im Gain-Knoten wird sie noch mit $\frac{1}{g^2}$ skaliert. Diese Nachricht wird dann am Schalter untersucht und es wird entschieden ob es sich um einen stimmhaften oder stimmlosen Abschnitt handelt. Die Nachricht $\mu_s(s_i)$ vom Schalter berechnet sich laut [4]:

$$\mu_s(s_i) = \mathcal{N}(m_{x_i} - m_y, V_{x_i} + V_y, s_i)\big|_{s_i=0} \tag{4.3}$$

$\mu_s(s_i)$ ist eine diskrete Nachricht, es werden die Gaußverteilungen der drei Schalterzustände um m_y nach links verschoben und an der Stelle Null ausgewertet. Somit erhalten wir die Wahrscheinlichkeitsverteilungen $P(s_i)$. Zur Entscheidungsfindung wird auch die Varianz der Nachricht $\mu_y(y)$ am Schalterausgang zur Varianz der einzelnen Zustände addiert. Je größer diese Varianz ist, umso weiter und flacher werden die drei Verteilungen (Abbildung 4.2 und 4.3) und somit werden die Wahrscheinlichkeitsverteilungen nicht mehr so klar für einen Zustand ausfallen. Dies wirkt sich in Folge auf das geschätzte Anregungssignal aus, da sich

[2] **A**dditive **W**hite **G**aussian **N**oise

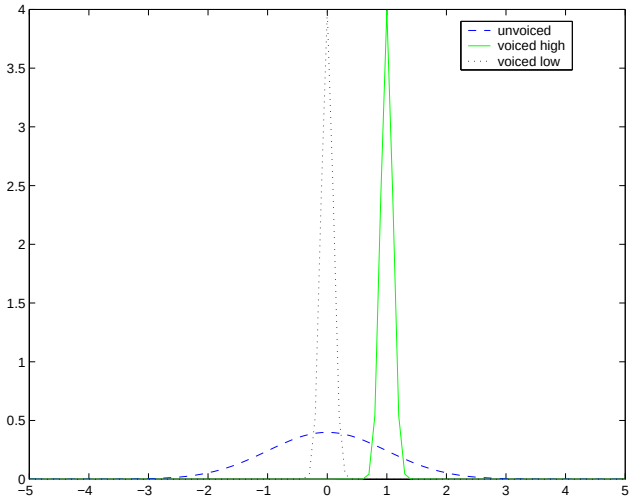


Abbildung 4.2: Illustration der Varianz der drei Zustände x_i am Schalter: stimmhafter Signalabschnitt $V_{x_0} = V_{x_1} = 0.01, m_{x_0} = 1$ bzw. $m_{x_1} = 0$ und stimmloser Abschnitt mit $V_{x_2} = 1, m_{x_2} = 0$.

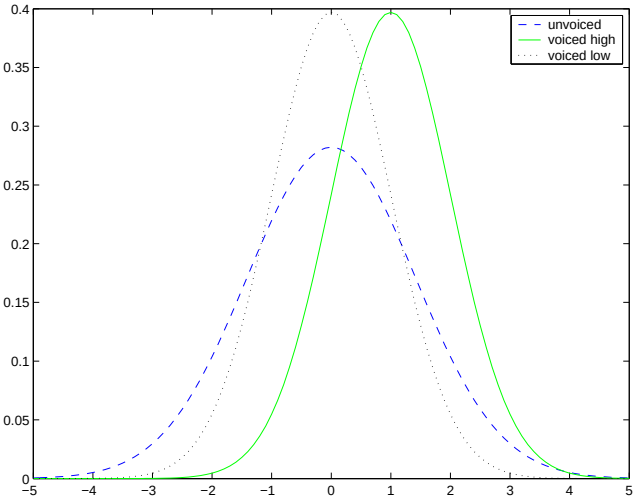


Abbildung 4.3: Varianzkurven am Schalter bei der Entscheidungsfindung mit verrauschtem Eingangssignal ($V_{Noise} = 1$): stimmhafter Signalabschnitt $V_{x_0} = V_{x_1} = 1.01, m_{x_0} = 1$ bzw. $m_{x_1} = 0$ und stimmloser Abschnitt mit $V_{x_2} = 2, m_{x_2} = 0$. Die drei Kuven überschneiden sich bei größerer Varianz immer mehr und damit wird beim Vergleichen mit m_y die Entscheidung für eine bestimmte Schalterstellung immer wager.

dieses aus folgenden Gleichungen berrechnet:

$$m_y = \mu_s(s_0)m_{x_0} + \mu_s(s_1)m_{x_1} + \mu_s(s_2)m_{x_2} \tag{4.4}$$

$$\begin{aligned} V_y = {} & \mu_s(s_0)\big(V_{x_0} + (m_y - m_{x_0})^2\big) + \mu_s(s_1)\big(V_{x_1} + (m_y - m_{x_1})^2\big) + \\ & + \mu_s(s_2)\big(V_{x_2} + (m_y - m_{x_2})^2\big). \end{aligned} \tag{4.5}$$

Wie man in den Gleichungen 4.4 sieht, wird der Mittelwert des Anregungssignal aus der Summe der gewichteten Mittelwerte der drei Zustände berechnet. Nur bei genauer Entscheidung zwischen den Zuständen wird die Wahrscheinlichkeitsverteilung für einen der drei dominant sein. Wenn alle anderen Randdichten im FFG berechnet sind, ist der Summe-Produkt Algorithmus abgeschlossen, und die Randdichte zwischen Additions- und Gleichheitsknoten in Abbildung 4.1 sollte eine möglichst gute Schätzung des unverrauschten Sprachsignals sein.

4.2 LPC-Koeffizienten

Herkömmliche LPC-Modelle für Sprache gehen von der Annahme aus, dass das Signal innerhalb des Analyseframes stationär ist. Sprachsignale sind in der Realität aber zeitvariant, d.h. die LPC-Koeffizienten werden sich über die Zeit verändern. In unserem FFG wird dies durch die Gleichheitsknoten zwischen den Filterknoten verhindert. Um einen kontinuierlichen Verlauf zu ermöglichen erlauben wir den Filterkoeffizienten über die Zeitachse einen *'random walk'*. Es wird über jeden Wert der Filtergewichte eine Varianzkurve gelegt, um so Abweichungen zuzulassen.

$$p(\underline{w}_2|\underline{w}_1) = \mathcal{N}(\underline{w}_2|\underline{w}_1, \mathbf{V}) \tag{4.6}$$

$$= \frac{1}{\sqrt{(2\pi\mathbf{V}}} exp\big(-\frac{1}{2}(\underline{w}_1 - \underline{w}_2)^t\mathbf{V}^{-1}(\underline{w}_1 - \underline{w}_2)\big) \tag{4.7}$$

Abbildung 4.4 zeigt eine Implementierung dazu. Es wird im Additionsknoten bei jedem Filterkoeffizient die Varianz erhöht, damit können sich die einzelnen Koeffizienten innerhalb des im Apriori-Knoten durch die Varianz eingestellten Rahmen bewegen. In Abbildung 4.5 sieht man einen solchen möglichen Verlauf der Filtergewichte. Dieser Verlauf wurde bei Verwendung des Graphen als generatives Modell, also zur Erzeugung eines synthetischen Ausgangssignals, verwendet. Abbildung 4.6 zeigt dieses Signal. Es ist dies ein rein stimmhafter Abschnitt und man sieht, wie sich das Ausgangssignal mit den Filtergewichten mitverändert. Dieses Signal wird später noch als Eingangssignal zur Analyse des FFG in Kapitel 6

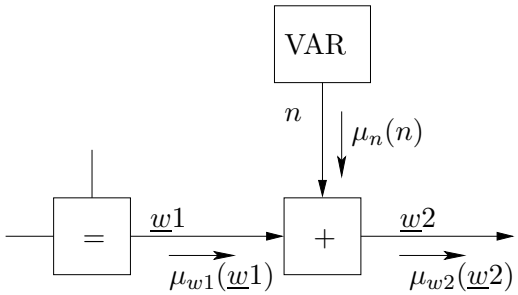


Abbildung 4.4: Erweiterung des FFG bei den Filtergewichten

verwendet, um so vergleichen zu können, ob der durch den Summe-Produkt Algorithmus berechnete Verlauf mit diesem übereinstimmt.

4.3 Gesamtes Modell

Der Faktor-Graph aus Abbildung 3.2 wie er in [4] entwickelt wurde, hat sich nun um oben besprochenens Störmodell und der Nachbildung für variable Filterkoeffizienten erweitert. Abbildung 4.7 zeigt nun einen Ausschnitt von zwei Slices des gesamten Graphen. Mit Hilfe dieses Modells wurden dann sämtliche Simulationen und Berechnungen durchgeführt.

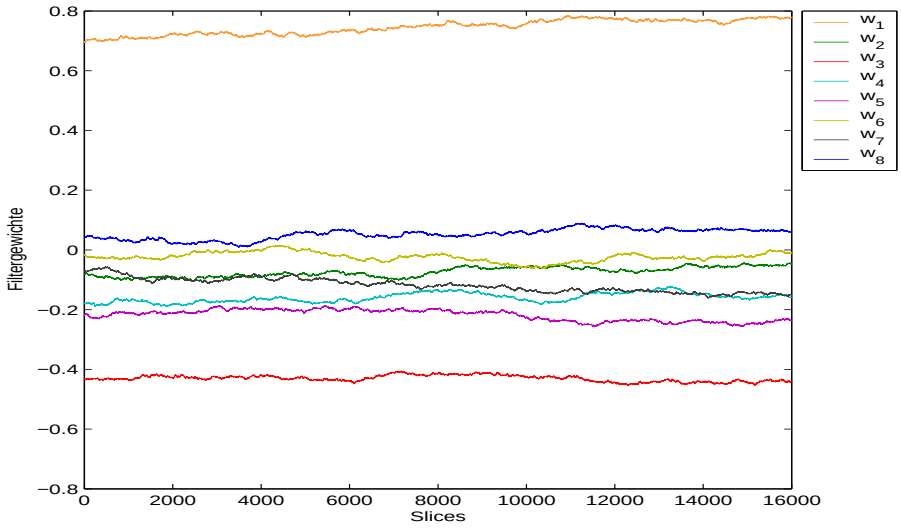


Abbildung 4.5: Durch die Addition der Varianz $V = 2 * 10^{-7}$ bei der Simulation verwendete Filterkoeffizienten.

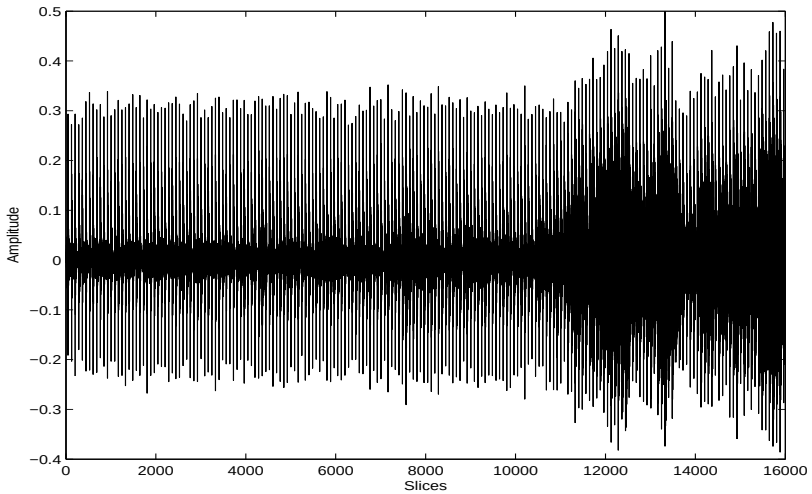


Abbildung 4.6: Bei der Simulation erhaltenes Ausgangssignal unter Verwendung der Koeffizienten aus Abbildung 4.5.

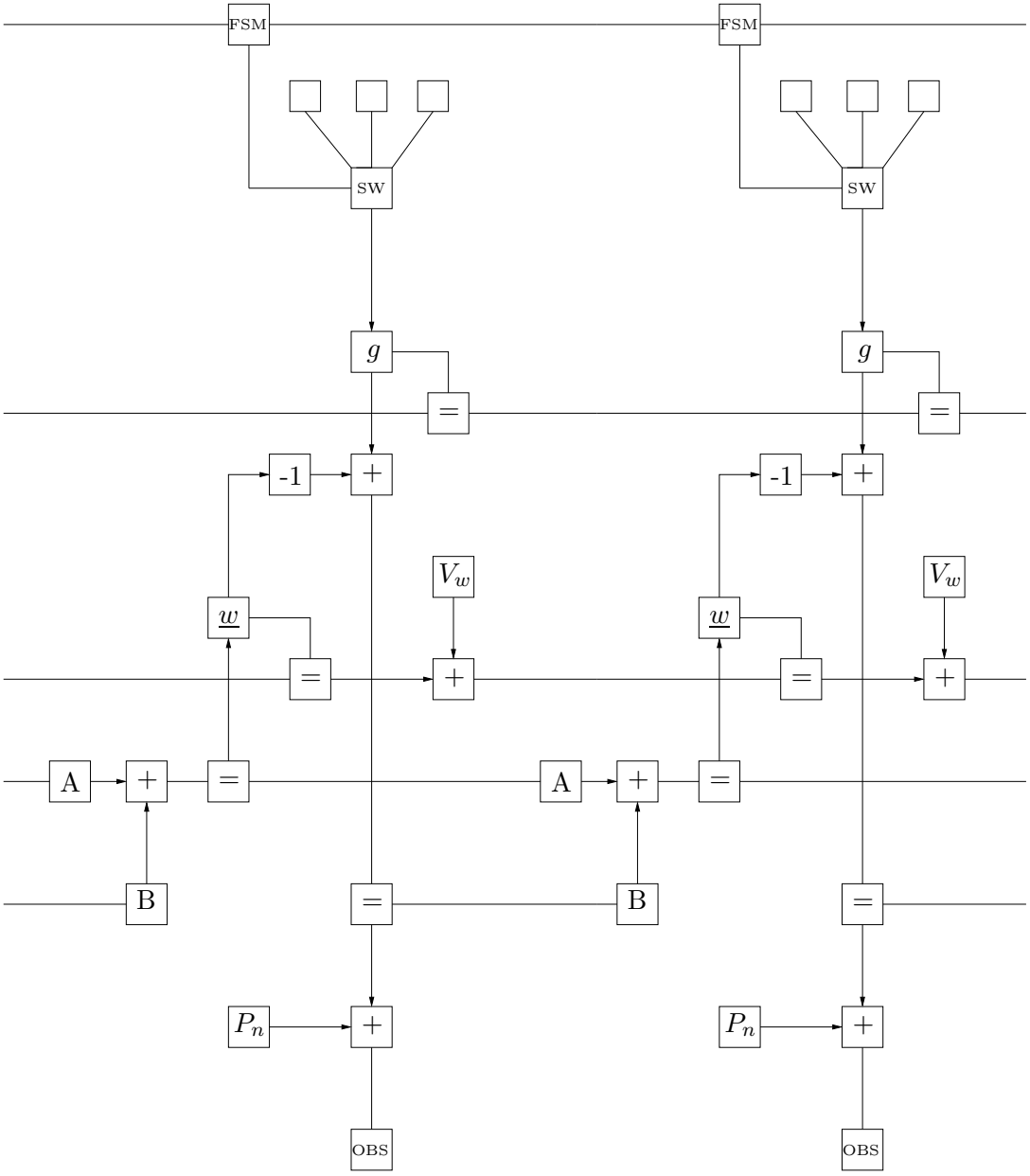


Abbildung 4.7: Gesamter FFG

Kapitel 5

Dynamischer Faktor-Graph

Um Faktor-Graphen für Verarbeitung von Audiosignalen in Echtzeit verwenden zu können, ist der nächste Schritt eine dynamische Erweiterung des Graphen. In diesem Kapitel werden Möglichkeiten einer solchen Dynamisierung besprochen.
Die Bezeichnung *'Dynamischer Graph'* bezieht sich eigentlich auf die Art der Realisierung mittels Software. Der Faktor-Graph als mathematisches Modell an sich bleibt unverändert und ist nicht dynamisch. Dies sollte immer bedacht werden, wenn vom dynamischen Aufbau gesprochen wird.

5.1 Prinzip

Wie bereits erwähnt, kann ein Sprachsample aus den vorhergegangen Werten geschätzt werden. Der Einfluss von weiter zurück liegenden Werten nimmt mit der Zeit immer mehr ab. Um eine kontinuierliche Verarbeitung eintreffender Audiosamples zu ermöglichen, muss der Faktor-Graph laufend mitwachsen. Um den Speicherbedarf und den Rechenaufwand dabei in Grenzen zu halten, können länger zurückliegende Slices gelöscht werden, die Anzahl der Slices zur Berechnung im FFG werden also konstant gehalten. In den folgenden Abschnitten werden zwei verschiedene Möglichkeiten zur Dynamisierung vorgestellt:

- FFG mit N Slices, der um i Slices erweitert wird
- FFG mit nur einem Slice

5.2 Dynamischer FFG mit N Slices

Der Faktor-Graph besteht also nun aus N Slices, die zur Berechnung herangezogen werden. Die Berechnung läuft nun folgendermaßen ab:
Der Faktor-Graph wird wie bisher mit N Slices aufgebaut und an den Rändern mit Apriori-Knoten abgeschlossen. Diese ersten N Samples werden mit dem Summe-Produkt Algorithmus berechnet, wobei mehrere Iteration gerechnet werden, da der FFG Schleifen besitzt. Diese N Samples sind auch die ersten N Ergebnisse unserer Schätzung des unverrauschten Sprachsignals. Nun beginnt der eigentliche dynamische Ablauf. Es werden i neue Slices angelegt und mit dem N-ten Slice verbunden, wobei die bereits berechneten Nachrichten an den aufgetrennten Kanten erhalten bleiben. Die Randknoten (Apriori-Knoten) müssen vorher vom Slice N getrennt werden und kommen an den Slice $N + i$, wobei sie ihre ursprünglich initialisierte Nachricht behalten. Damit die Anzahl der Slices konstant bleibt, werden die ersten i Slices gelöscht und die linken Apriori-Knoten mit dem nächsten Slice verbunden. Damit auch die Information von vorherigen Samples nicht verloren geht, werden die berechneten Nachrichten ebenfalls übernommen. So kann dann wieder der Summe-Produkt Algorithmus gestartet werden. Es wird gesamte Graph mehrere Male iteriert und die so neu berechneten i Samples ergeben die nächsten i gültigen Werte unserer Schätzung.
Abbildung 5.1 zeigt dieses Prinzip der Bewegung des Graphen über das Sprachsignal. Mit der Größe i lässt sich der Einfluss der alten Slices variabel gestalten und die Rechenzeit verringern. In welchem Verhältnis alte zu neuen Slices stehen sollen muss empirisch ermittelt werden. Je größer die Anzahl der Slices, die zur Berechnung in einem Durchgang verwendet werden, umso genauer haben sich die Filterkoeffizienten eingestellt.

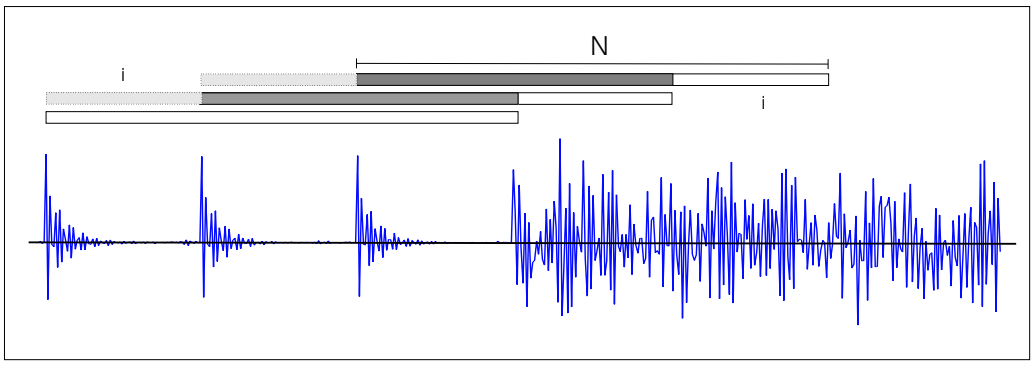


Abbildung 5.1: Prinzipielle Darstellung des Auf- und Abbaus des dynamischen Graphen.

5.2.1 Schedule des Graphen

Die Reihenfolge der Berechnungen nach den Update Rules aus Abschnitt 3.4 wurde für den Graphen wie folgt festgelegt:

1. Sämtliche **Apriori-Knoten** lassen sich problemlos im ersten Schritt berechnen.
2. Alle N **Eingangssamples** werden eingelesen, die Störvarianz addiert und der Gleichheitsknoten berechnet.
3. Die Eingangsnachricht wird über den **Additionsknoten** an den Anregungsteil weitergeleitet.
4. Der **Verstärkungsfaktor** wird bestimmt und über alle N Slices abgeglichen.
5. Die Nachricht wird am **Schalter** analysiert und die Entscheidungen für die drei Zustände an die **FSM** gegeben, welche dann von links nach rechts und retour alle Zustände durchrechnet.
6. Mit den so gefundenen Zuständen wird das **Schalterausgangssignal** und damit auch das Anregungssignal errechnet.
7. Bevor das Ausgangssignal berechnet wird, muss noch der **Filter** durchlaufen werden. Dazu wird die **SSM** vom ersten Slice bis zum letzten durchlaufen und wieder zurückgerechnet. Dann kann das Filter berechnet werden.
8. Die **Filterkoeffizienten** werden über alle Slices vor und zurück gerechnet, wobei diese jetzt nicht mehr zeitinvariant sind.
9. Jetzt kann die Ausgangsnachricht über den **Additionsknoten** berechnet werden.
10. Nach mehreren **Iterationen** sollten die Variablen im Graphen konvergieren.

5.3 FFG mit einem Slice

Die Berechnungen in den vorherigen Abschnitten sind zum Teil sehr rechenintensiv und nehmen viel Zeit in Anspruch. Folgende Überlegungen gehen von dem Ziel aus, die Berechnungen an Audiosignalen in Echtzeit vorzunehmen; ein Sample trifft ein, eines wird neu berechnet. Wir treffen die Einschränkung, dass zukünftige Werte, die durch Rückrechnung bis jetzt berücksichtigt wurden, nun nicht mehr in die Berechnungen miteinfließen.
Betrachten wir den FFG eines gesamten Slices als einen Knoten mit einem Eingangs- und

Ausgangsvektor, die die Kante zu dem vorherigen bzw. nächsten Slice bilden.

$$\mathbf{U_{N-1}} = \begin{pmatrix} \mu_{f_{N-1}}(f_{N-1}) \\ \mu_{g_{N-1}}(g_{N-1}) \\ \mu_{w_{N-1}}(w_{N-1}) \\ \mu_{s_{N-1}}(s_{N-1}) \\ \mu_{x_{N-1}}(x_{N-1}) \end{pmatrix} \qquad \mathbf{U_N} = \begin{pmatrix} \mu_{f_N}(f_N) \\ \mu_{g_N}(g_N) \\ \mu_{w_N}(w_N) \\ \mu_{s_N}(s_N) \\ \mu_{x_N}(x_N) \end{pmatrix}$$

Die beiden Vektoren $\mathbf{U_{N-1}}$ und $\mathbf{U_N}$ enthalten die einzelnen Nachrichten in Vorwärtsrichtung. Durch das Weglassen der Rückrechnung ist der FFG schleifenfrei und damit die Nachrichten in den Randvektoren unabhängig geworden. Der Schedule des Summe-Produkt-Algorithmus ändert sich dahingehend, dass der Graph nun *'straight forward'* berechnet werden kann und keine Iterationen mehr benötigt werden.
Für die Dynamisierung bedeutet dies, dass wir den Faktor-Graphen auf einen Slice reduzieren können. Dieser neue Graph in Abbildung 5.2 kann nun mit minimalem Zeitaufwand berechnet werden. Ein neuer Slice wird nur aus den vorherigen Werten berechnet, da bei der Dynamisierung die Nachrichten erhalten bleiben. Einen längeren Einschwingvorgang muss man jedoch in Kauf nehmen.

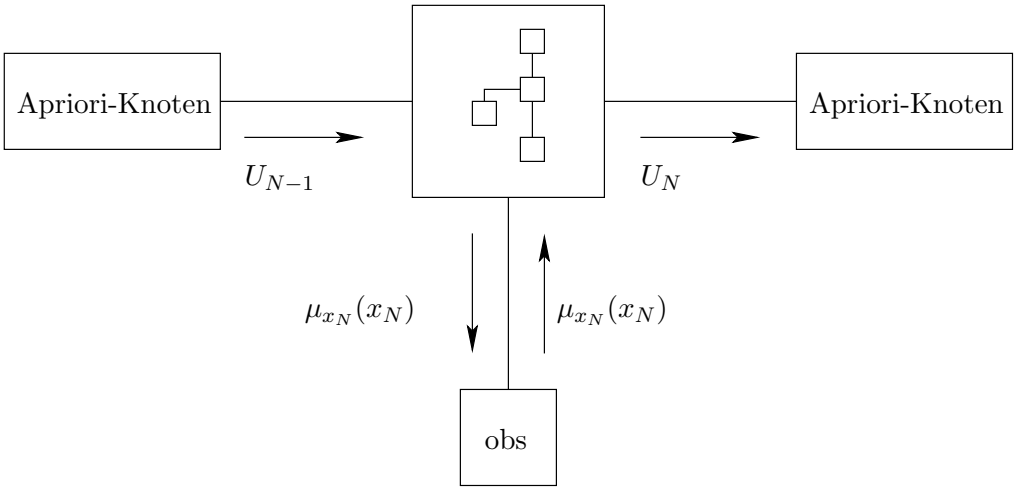


Abbildung 5.2: FFG optimiert, ohne Rückrechnung bestehend aus nur einem Slice.

5.3.1 Schedule des Graphen

Der Schedule für das reine *'Forward Passing'* lässt sich einfach ableiten. Man beginnt wieder bei den Apriori-Knoten, wobei die auf der rechten Seite nicht berechnet werden müssen, da wir keine Information von zukünftigen Slices zulassen. Anschließend durchläuft man den Graphen von unten nach oben und wieder zurück. Dabei ergibt sich die Reihenfoge automatisch; man berechnet immer den nächsten Knoten, an dem alle benötigten Nachrichten gültig anliegen. Nachrichten, die nicht in die Berechnung einfließen haben,da sie beim Aufbau des Graphen mit großer Varianz initialisiert werden, keinen Einfluß auf die Berechnung. In Abbildung 5.3 ist der Schedule eines Slices durchnummeriert.

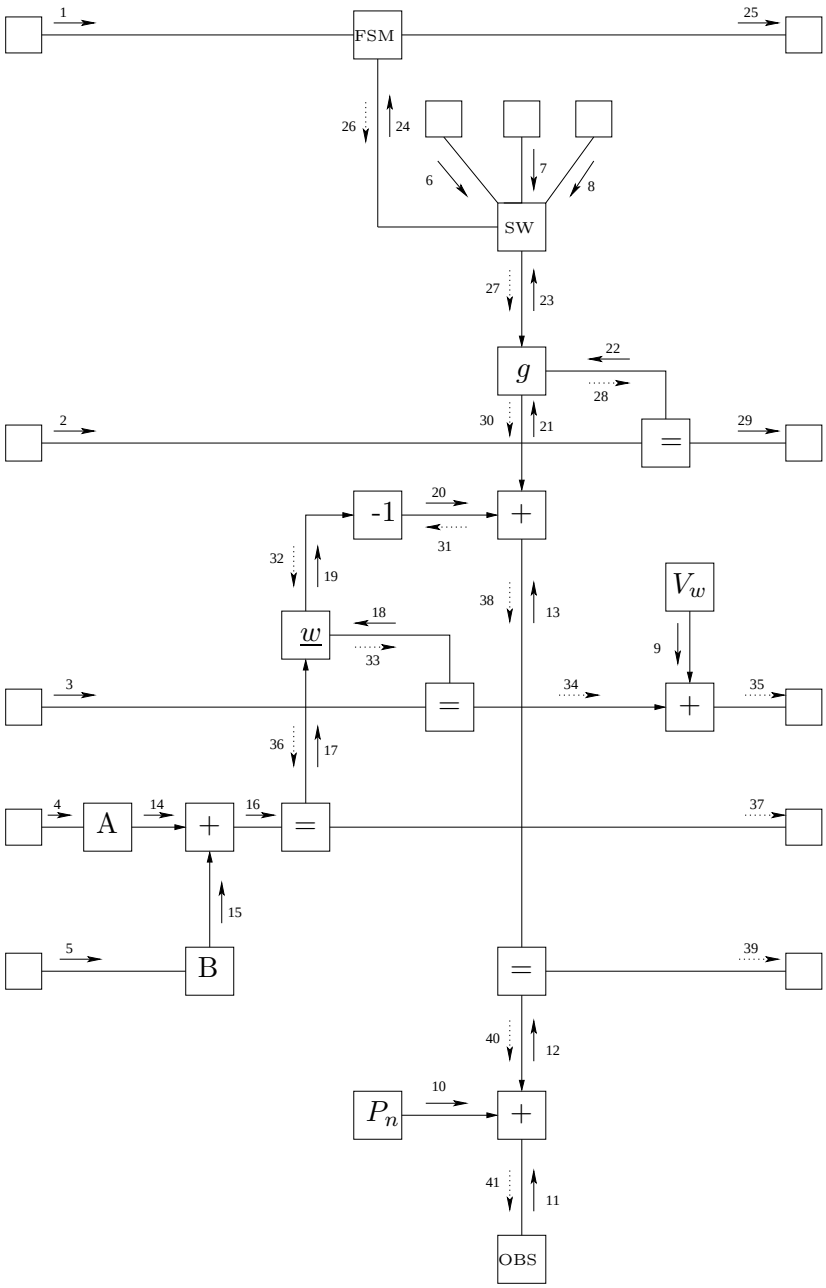


Abbildung 5.3: Schedule des FFG bestehend aus nur einem Slice.

Kapitel 6

Resultate

In diesem Kapitel sollen die Modellerweiterungen und Änderungen im Graphen getestet werden. Die Eingangssignale dazu wurden, wie in Abschnitt 2.5 besprochen, mittels des FFG selbst generiert. Da die Entwicklungsumgebung von MATLAB auf C++ gewechselt wurde, werden zuerst allgemeine Untersuchungen am Graphen unternommen.
Durch die Einführung der Dynamisierung sind wir in der Lage längere Signale mit größerer Impuslsperiodendauer, die realen Sprachsignalen schon näher kommen, zu testen. Es wird an den verschiedenen Arten der dynamischen Berechnung getestet, wie sich ein Störsignal auswirkt und ob ein solches erfolgreich eliminiert werden kann. Weiters soll untersucht werden, wie gut die Modellierung der variablen Filterkoeffizienten die Wirklichkeit abbilden kann.

6.1 FFG des Srachmodells

Im folgenden Abschnitt werden im Faktor-Graph folgende Punkte analysiert:

- **Einstellung der Filterkoeffizienten**: Dem FFG wird in den Randbedingungen bei den Simulationen keine Vorinformation über die Filtergewichte gegeben. Es wird untersucht ob sich diese im Graphen richtig einstellen und welche Rolle dabei die Signallänge spielt.
- **Schedule des Summe-Produkt Algorithmus**: Die Reihenfolge der Berechnung der einzelnen Nachrichten im Graph spielt eine wesentliche Rolle für die Anzahl der notwendigen Iterationen und ob die Variablen im Graph konvergieren.
- **Konvergenzverhalten des Graphen**: Stellen sich die internen Größen im FFG auf einen fixen Wert ein? Diese Frage stellt sich, da durch die vorhandenen Schleifen

im Graphen der Summe-Produkt Algorithmus iterativ abgearbeitet werden muss und somit die Nachrichten an den Knoten mehrmals aktualisiert werden.

Da die Entwicklungsumgebung geändert wurde, wurde der Graph in C++ neu aufgebaut und mit den ersten Tests auch die Programmierung der einzelnen Funktionen in der neuen Umgebung getestet. Um für die folgenden Simulationen Eingangssignale zu erstellen, wurde der FFG wie in Abschnitt 2.5 erklärt als generatives Modell verwendet. Das hat den Vorteil, dass wir die Eigenschaften des Eingangssignals kennen und somit überprüfen können, ob die mittels FFG berechneten Werte stimmen.

6.1.1 Filtergewichte

Wir verwenden ein Eingangssignal mit einer Impulsperiode von $P = 10$, Impulsamplitude $\kappa_a = 1$ für stimmhaften Abschnitt und weißes Rauschen mit Leistung $P_{uv} = 0.1$ für stimmlosen Abschnitt. Der Verstärkungsfaktor und die Filtergewichte werden durch die Gleichheitsknoten über die jeweils verwendeten Slices gemittelt, im Zweig der SSM wird im Schedule festgelegt, dass nicht zurückgerechnet wird.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
0.697	-0.08	-0.434	-0.181	-0.218	-0.018	-0.07	0.041

Tabelle 6.1: Filtergewichte mit denen das Eingangssignal generiert wurde. $w_0 = 1$ ist durch die Filterstruktur vorgegeben ([4]).

Erste Simulation, 100 Slices

Die dabei ermittelten Filterkoeffizienten stellt Abbildung 6.1 dar. Man sieht, dass sich die Filtergewichte noch nicht optimal einstellen. Die beiden weiteren Simulationen zeigen, dass hier die Berechnungslänge von 100 Slices zu kurz war.

Zweite Simulation, 500 Slices

Das Eingangssignal ist jetzt fünf mal so lang, Abbildung 6.2 zeigt die somit errechneten Filtergewichte. Hier haben wir bereits eine gute übereinstimmung mit den Filtergewichten aus Tabelle 6.1.1.

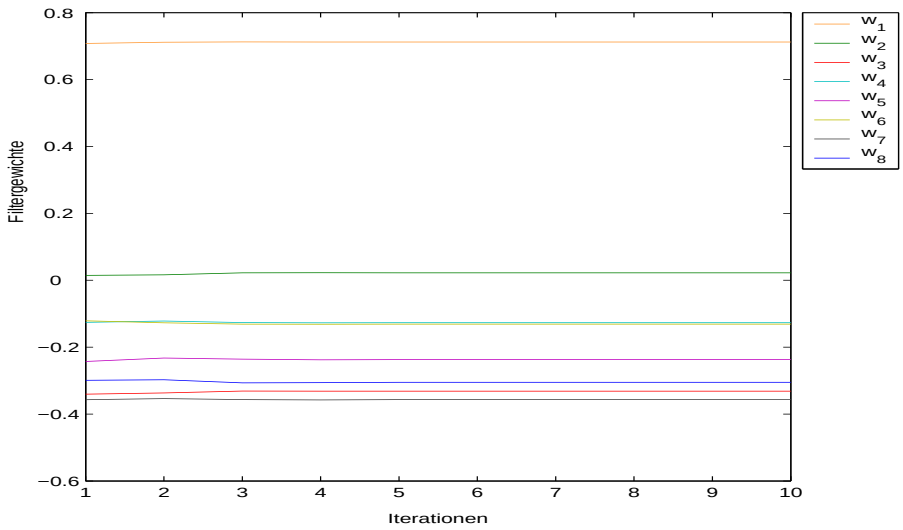


Abbildung 6.1: Resultat der 1.Simulation: Einschwingverhalten der vom FFG berechneten Filterkoeffizienten w_i.

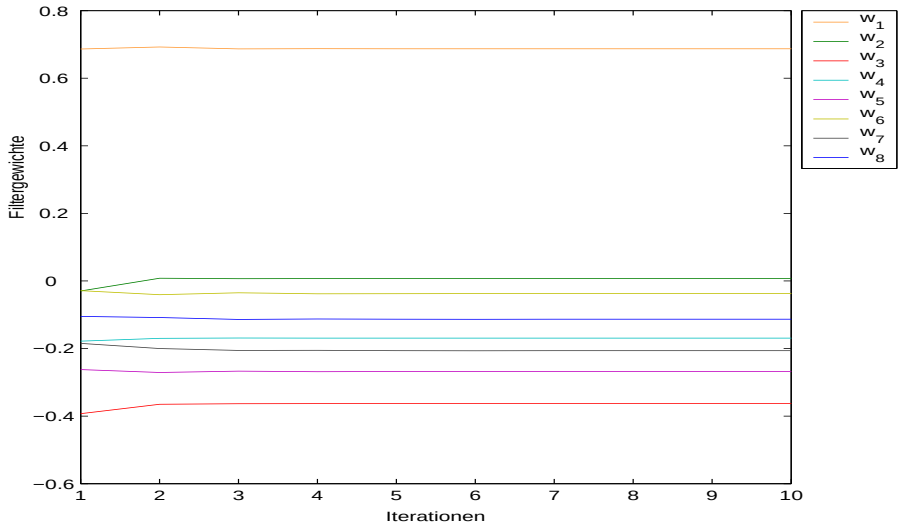


Abbildung 6.2: Der Vergleich zur 1.Simulation: Filterkoeffizienten w_i über 500 Slices ermittelt.

Dritte Simulation, dynamischer Graph mit 500 Slices

Wenn wir nun als dritten Versuch den dynamischen Faktor-Graphen mit N Slices verwenden betätigt sich obiges Ergebnis. Es wird das gleiche Eingangssignal verwendet und 100 Slices als Berechnungsbreite festgelegt. Der FFG erweitert sich jeweils um 10 Slices, die Information aus den früheren Slices wird immer mitübernommen. In Abbildung 6.3 ist zu erkennen, wie sich die Filterkoeffizienten über die Zeit immer genauer einstellen. Zusammenfassend

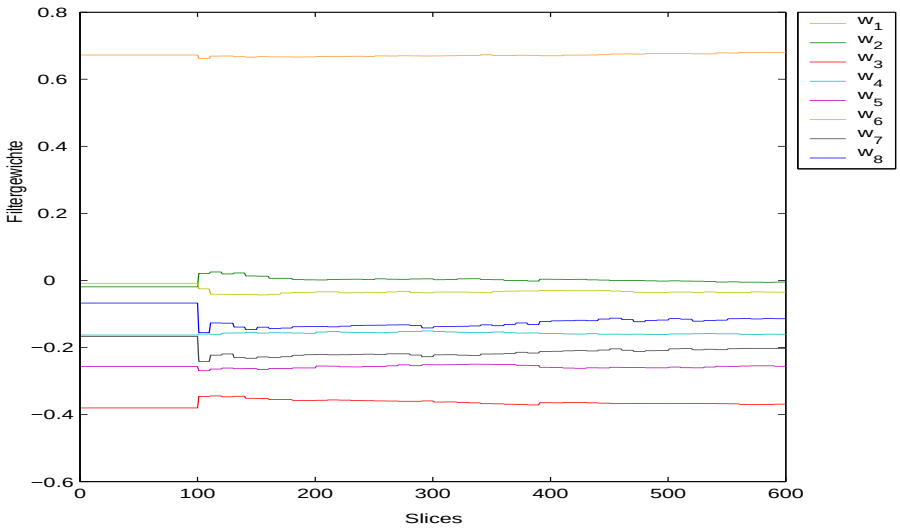


Abbildung 6.3: Verlauf der Filterkoeffizienten w_i bei Verwendung des dynamischen FFG.

kann nach obigen Simulationen gesagt werden, dass der Faktor-Graph bei Verwendung einer größeren Anzahl von Slices die Filterkoeffizienten immer genauer einstellt. Wird der FFG dynamisch gemacht, werden Filtergewichte praktisch über die gesamte Signallänge gemittelt und stellen sich auch bei Verwendung von weniger Slices während eines Berechnungsdurchgangs mit der Zeit genau ein.

6.1.2 Einfluss des Schedule auf Iterationen bzw.Konvergenzverhalten

Wie in Kapitel 2 erklärt legt man mit dem Schedule die Reihenfolge fest, mit der die Nachrichten an den einzelnen Knoten im FFG aktualisiert werden. Man muss darauf achten, dass bei einem Update einer Nachricht am Knoten gültige und aktuelle Nachrichten anliegen, da sonst die Randdichten im Graphen nicht konvergieren werden. Abbildung 6.1 und 6.4 zeigen die gleiche Simulation mit unterschiedlichen Schedule. Es wurde dabei der gesamte FFG in alle Richtungen berechnet.

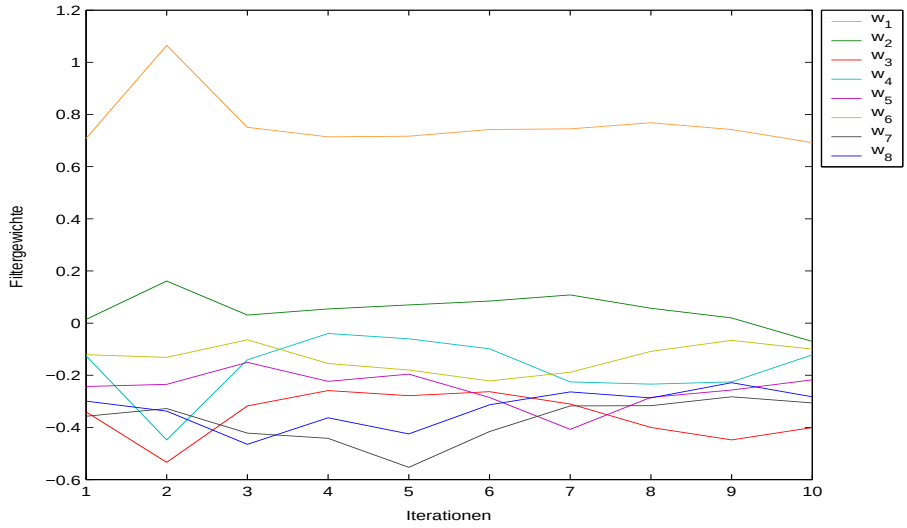


Abbildung 6.4: Beispiel für einen schlechten Schedule; die Filtergewichte konvergieren nicht.

6.1.3 Konvergenzverhalten

Wie die beiden Abbildungen 6.1 und 6.2 zeigen, konvergieren die Filtergewichte schon nach drei Iterationen. Man kann also den Einschwingvorgang nach drei Iteration als abgeschlossen ansehen. Wichtige Voraussetzung dafür ist ein sinnvoller Schedule. In Abschnitt 2.3 haben wir den schleifenfreien Graphen besprochen. Da in diesem Fall die Berechnung der Randdichten in einem Durchgang möglich ist, sind keine Iterationen notwendig.

6.2 Dynamischer FFG mit N Slices

Die Berechnungen der folgenden Simulationen erfolgen jetzt mit dem dynamischen Graphen, da wir uns in Richtung reale Audiosignale vorarbeiten. Wir verwenden eine Impulsperiode von $P = 80$, das entspricht nach Formel 6.1 einer Frequenz von 200 Hz bei einer Abtastfrequenz von 16kHz.

$$T_0 = \frac{1}{Grundfrequenz} \qquad T_s = \frac{1}{Abtastfrequenz} \qquad P = \frac{T_0}{T_s} \tag{6.1}$$

Um Signale der Länge von einer Sekunde zu berechnen werden dann 16000 Slices benötigt. Dies wird bei Verwendung eines statischen FFG zu rechenintensiv und zeitaufwändig, da durch die Schleifen mehrere Iterationen über alle Slices gerechnet werden. Dies ist nicht notwendig, da der Einfluß weiter zurückliegender Werte verschwindet.

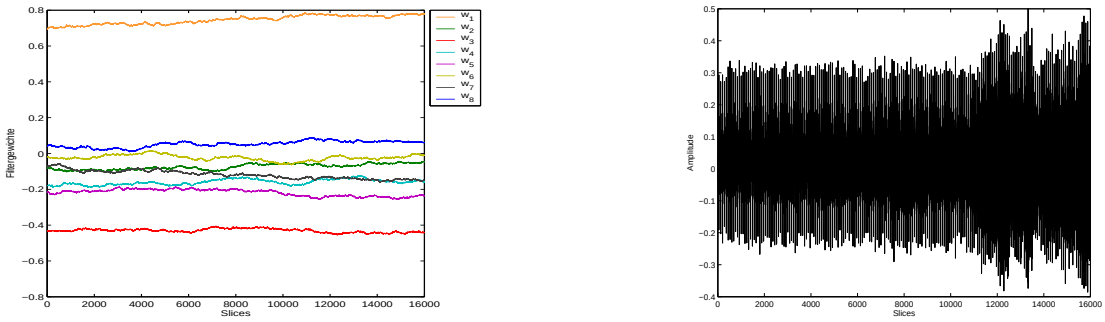


Abbildung 6.5: Durch die Addition der Varianz $V = 2 \cdot 10^{-7}$ bei der Simulation verwendete Filterkoeffizienten (links) und erhaltenes Ausgangssignal (rechts).

6.2.1 Variable Filterkoeffizienten

In diesem Abschnitt wird der FFG mit der Modifizierung an den Filtergewichten aus Abschnitt 4.2 getestet. Untersucht wird, ob der Graph den Verlauf der Filtergewichte wieder richtig erkennt. Das Signal aus Abbildung 6.5(rechts) verwenden wir nun als Eingangssignal des FFG. Die Berechnung erfolgt mit einem dynamischen Graphen der Länge $N = 300$ und $i = 100$ neuen Slices. Nach Abschluss des Summe-Produkt-Algorithmus interessiert uns der Verlauf der Filtergewichte über die Zeit. Das Ergebnis zeigt Abbildung 6.6. Man sieht, dass der FFG bei der Analyse des Signals den Verlauf der Filterkoeffizienten, wie er zur Erzeugung des Signals verwendet wurde (Abbildung 6.5 (links)), erkennt, auch wenn sich die Koeffizienten in diesem Fall nur in sehr engen Bahnen bewegen.

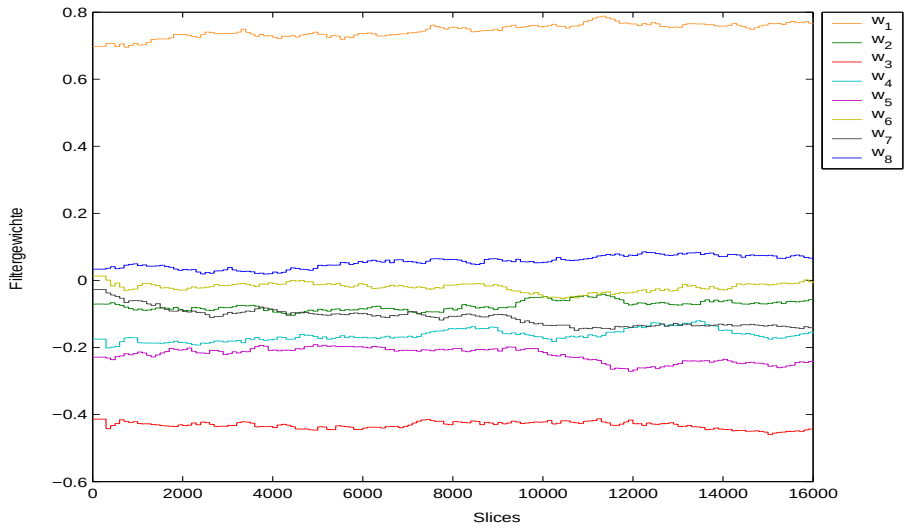


Abbildung 6.6: Verlauf der vom FFG berechneten Filterkoeffizienten w_i mit Eingangssignal aus Abbildung 6.5(rechts).

6.3 Dynamischer FFG mit einem Slice

In diesem Abschnitt wird der in Kapitel 5 erklärte Faktor-Graph getestet. Die Simulation wird mit einem Signal der Länge 16000 Slices mit einer Impulsperiode $P = 80$, Impulsamplitude $\kappa = 1$ und Varianz $V_v = 0.0001$ für den stimmhafte Abschnitt und einer Varianz $V_{uv} = 0.1$ für stimmlose Abschnitte, durchgeführt. Die Varianz einer Störung und der Filterkoeffizienten wird dann in den verschiedenen Simulationen variiert.
Von Interesse ist:

- **Einschwingverhalten des Graphen**: Da jetzt nicht mehr über mehrere Slices gemittelt, ist zu erwarten, dass der Graph etwas Zeit zum Einschwingen benötigt.
- **Filtergewichte**: Wie genau stellen sich die Filtergewichte ein, wenn nur mehr in Vorwärtsrichtung gerechnet wird? Wird der zeitliche Verlauf bei variablen Filterkoeffizienten nachvollzogen?
- **Signalklassifizierung**: Um ein Signal wieder zu rekonstruieren, müssen am Schalter die stimmhaften und -losen Abschnitte richtig erkannt werden damit die FSM synchronisiert wird.
- **Störgeräuschbefreiung**: Wie wirkt sich ein Störsignal am Eingang aus und kann es

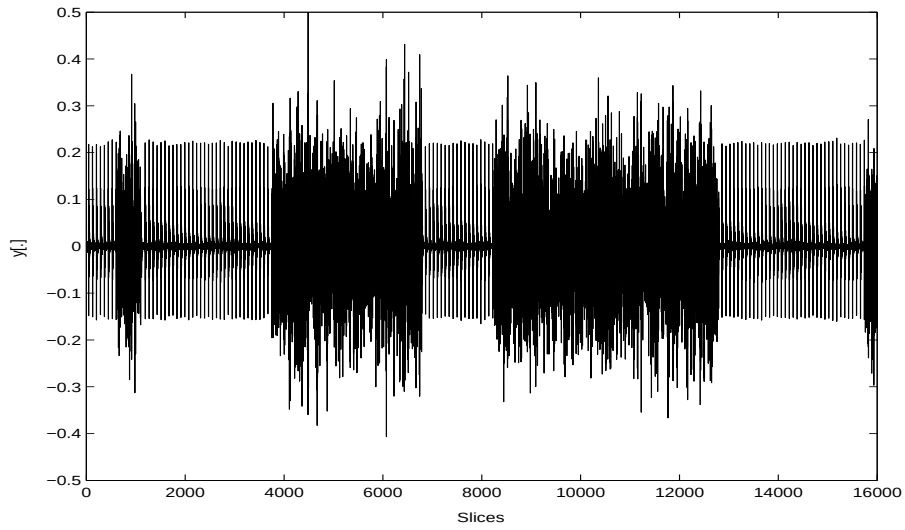


Abbildung 6.7: Generiertes unverauschtes Eingangssignal y[.]

erfolgreich eliminiert werden?

6.3.1 FFG ohne Störung, dynamisch mit einem Slice

Zunächst untersuchen wir den dynamischen FFG ganz allgemein. Wie in Abschnitt 5.3 besprochen, besteht der Graph nur aus einem Slice und den jeweiligen Apriori-Knoten am Rand. Die Berechnung der Randdichten erfolgt nur in Vorwärtsrichtung, es wird ein unverrauschtes Signal (Abbildung 6.7) am Eingang angelegt.
Die Filtergewichte (Abbildung 6.8) stellen sich nach einem Einschwingvorgang von ca. 2500 Slices sehr exakt ein. Die Einschwingdauer begründet sich daraus, dass der FFG nur aus dem einen Slice bei der Berechnung besteht und nicht wie im vorigen Abschnitt aus N Slices, über die gemittelt wurde. Durch diese *'straight forward'* Strategie beim Summe-Produkt Algorithmus wird mehr Zeit benötigt, bis der richtige Verstärkungsfaktor gefunden wird und dann auch im Anregungsteil die richtigen Signalabschnitte identifiziert werden.

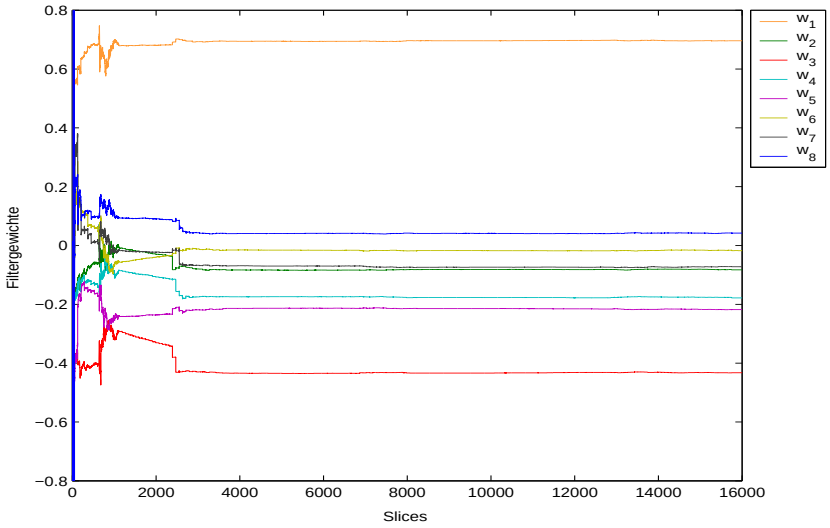


Abbildung 6.8: Berechnete Filtergewichte. Der Sprung um den Slice 2500 herum passiert dadurch, dass das Anregungsmodell erst ab dann die drei Zustände richtig erkennt.

Betrachten wir noch weitere Nachrichten im FFG. Die Nachricht am Verstärkereingang zeigt Abbildung 6.9. Diese gleicht dem Ausgangssignal des Anregungsmodells und wurde aus dem Eingangssignal geschätzt. Diese Nachricht wird jetzt in jedem Slice am Schalter mit den drei möglichen Zuständen verglichen und die jeweiligen Wahrscheinlichkeiten der Übereinstimmung (Abbildung 6.10) vom Kontrolleingang zur FSM geschickt. Wenn man die beiden Abbildungen 6.9 und 6.10 vergleicht, sieht man, dass erst wenn sich der Verstärkungsfaktor richtig eingestellt hat und damit die Amplitude der Impulse groß genug ist, wird der Zustand *'voiced high'* auch vom Schalter erkannt. Der Kontrollausgang der Finite State Machine (Abbildung 6.11) liefert dann auch eine ganz eindeutige Wahrscheinlichkeitsverteilung der drei Zustände.

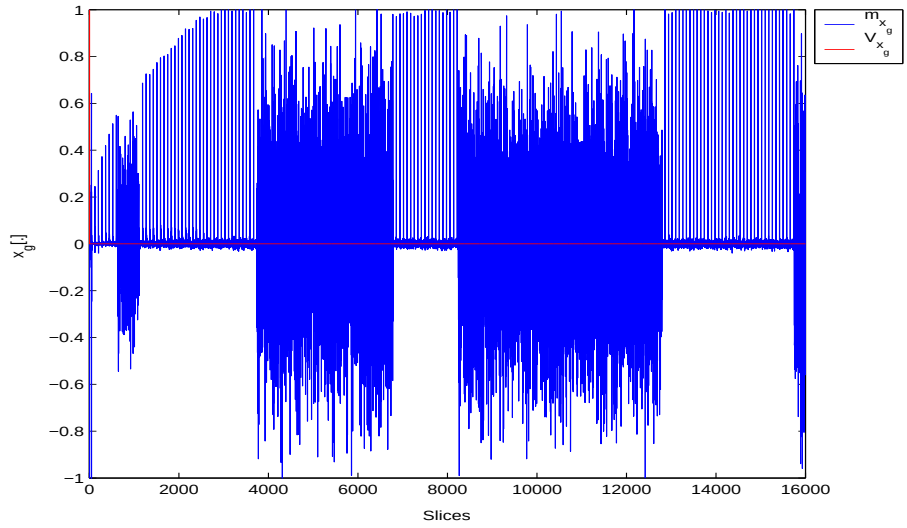


Abbildung 6.9: Mittelwert der Nachricht, welche vom Verstärkerknoteneingang zum Schalterausgang geht. Wenn die Amplitude der Impulse groß genug ist, wird der *'voiced high'* Zustand vom Schalter richtig erkannt.

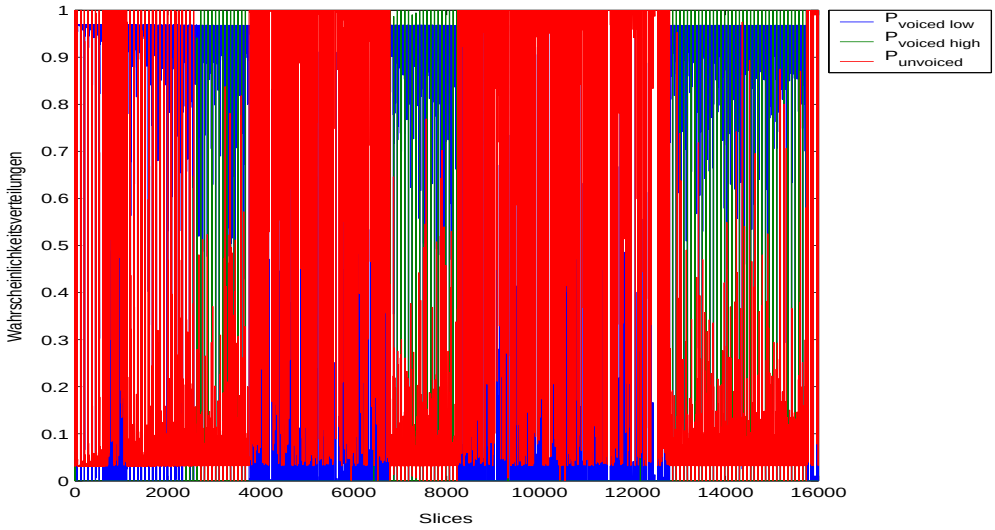


Abbildung 6.10: Wahrscheinlichkeitsverteilung der drei Zustände am Kontrolleingang des Schalters.

6.3.2 Störgeräuschbefreiung

Bei den nächsten Simulationen fügen wir das Störquellenmodell am Eingang ein. Es wird ein weißes Gauß'sches Rauschen mit Mittelwert $m = 0$ und einer Leistung $P_n = V_n$ verwendet. Wir wollen das Verhalten bei verschieden großen Rauschleistungen untersuchen. Die Varianz des stimmhaften Teils am Schalter muss etwas angepaßt werden, da sonst der Bereich in dem *'voiced high'* bewertet wird zu schmal ist. Bei folgenden Simulation ist $V_{voiced} = 0.001$.

Erste Simulation, $P_n = 0.001$

Beginnen wir mit einer sehr kleinen Störung. Abbildung 6.12 zeigt das verrauschte Eingangssignal. Die Filtergewichte (Abbildung 6.13) stellen sich nach dem Einschwingen wieder sehr genau ein und ändern sich nur sehr gering über die Zeit. Abbildung 6.14 zeigt die Wahrscheinlichkeitsverteilung der drei Schalterzustände am Kontrollausgang der Finite State Machine und Abbildung 6.15 verdeutlicht die Ausgangsnachricht des Anregungsmodells. Für die stimmhaften Abschnitte ergibt sich der Impulszug mit Periode $P = 80$, $m = 1$ und der kleinen Varianz $V_v = 0.001$, die stimmlosen Abschnitte werden mit $m = 0$ und der grösseren Varianz $V_{uv} = 0.1$ erzeugt. Abbildung 6.16 zeigt das somit geschätzte unverrauschte Ausgangssignal.

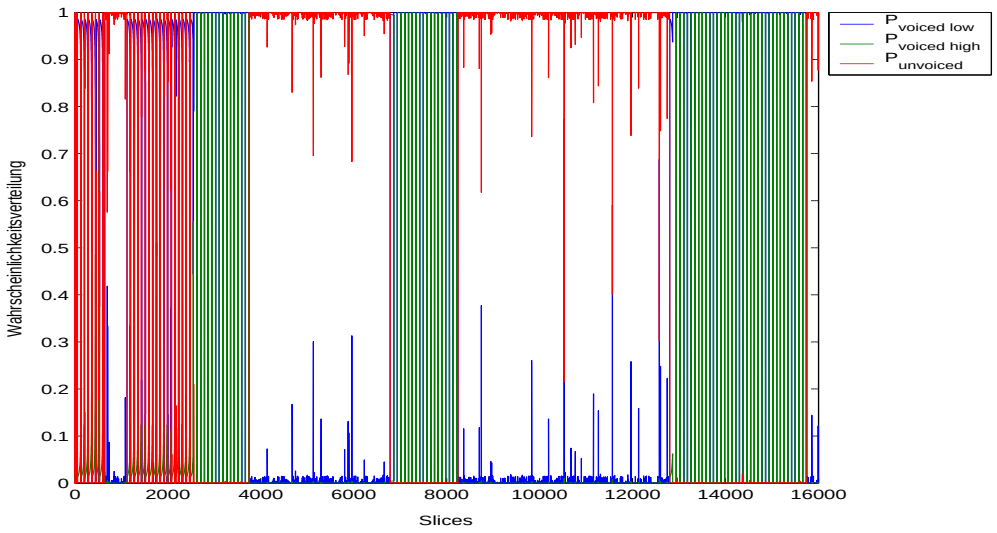


Abbildung 6.11: Wahrscheinlichkeitsverteilung der drei Zustände am Kontrollausgang der FSM. Wenn die FSM synchronisiert ist, werden die Zustände eindeudig wiedererkannt

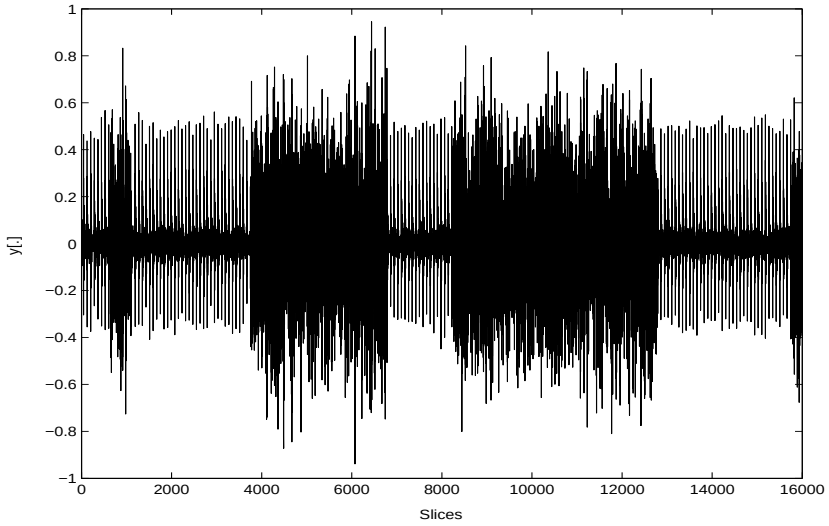


Abbildung 6.12: Verrauschtes Eingangssignal mit $P_n = 0.001$.

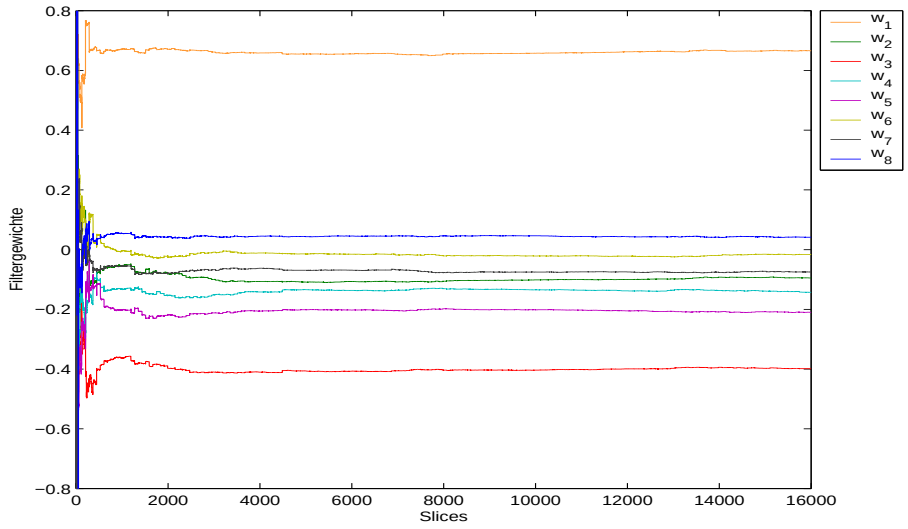


Abbildung 6.13: Die vom FFG gefundenen Filtergewichte stellen sich schneller ein, da die FSM schneller die richtigen Zustände entscheidet..

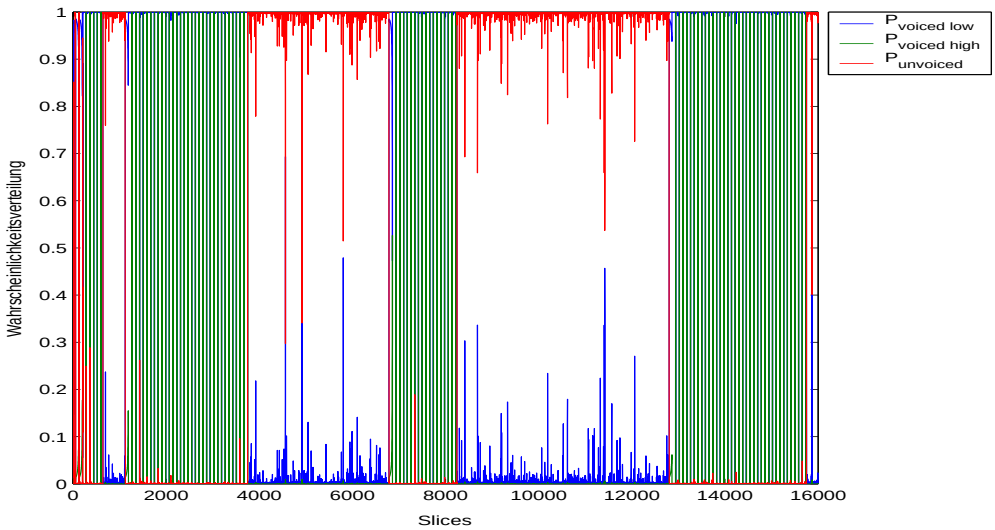


Abbildung 6.14: Wahrscheinlichkeitsverteilung am Steuerausgang der FSM, die den Schalter steuert. Es dominiert immer einer der drei Zustände.

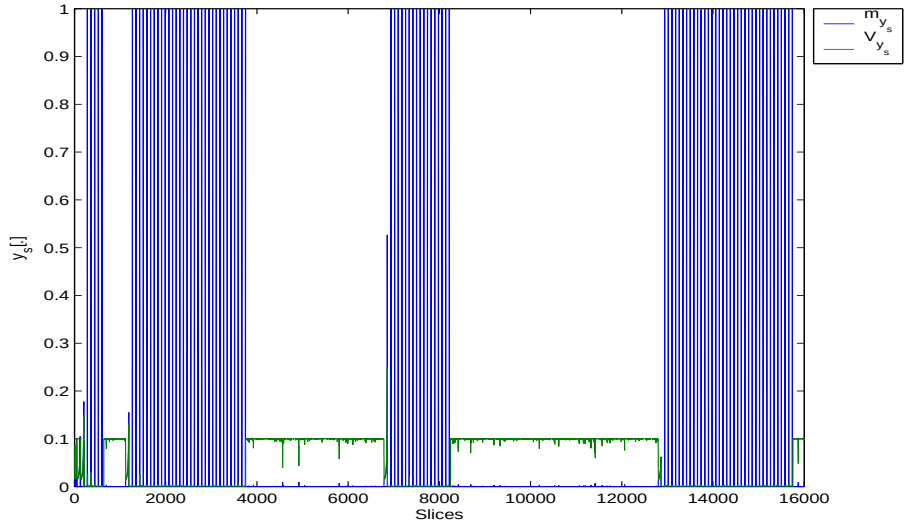


Abbildung 6.15: Mittelwert und Varianz der Nachricht am Schalterausgang. Für stimmhafte Abschnitte wird ein Impulszug mit der Periode P erzeugt, bei den stimmlosen Abschnitten ist der Mittelwert $m = 0$ und die Varianz etwas höher.

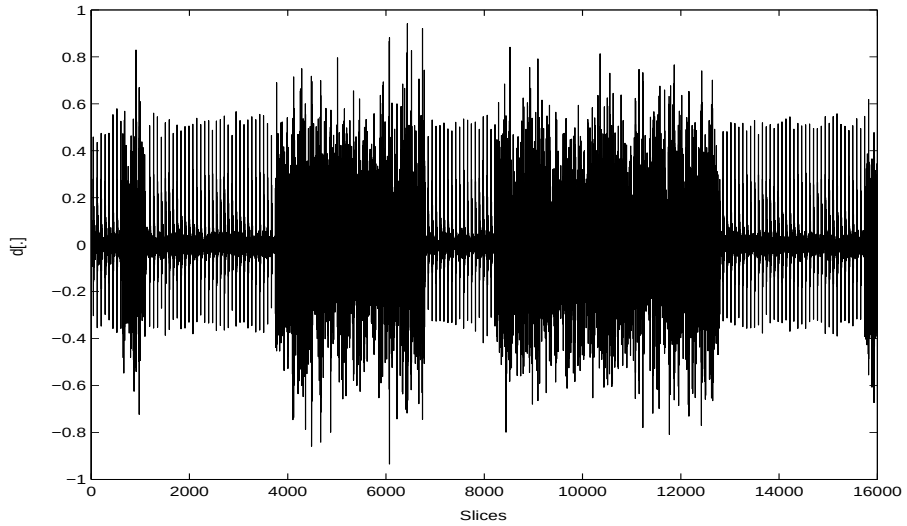


Abbildung 6.16: Die Schätzung d[.] des Eingangssignals bildet das unverrauschte Eingangssignal ab, wobei die Rauschleistung $P_n = 0.001$ war.

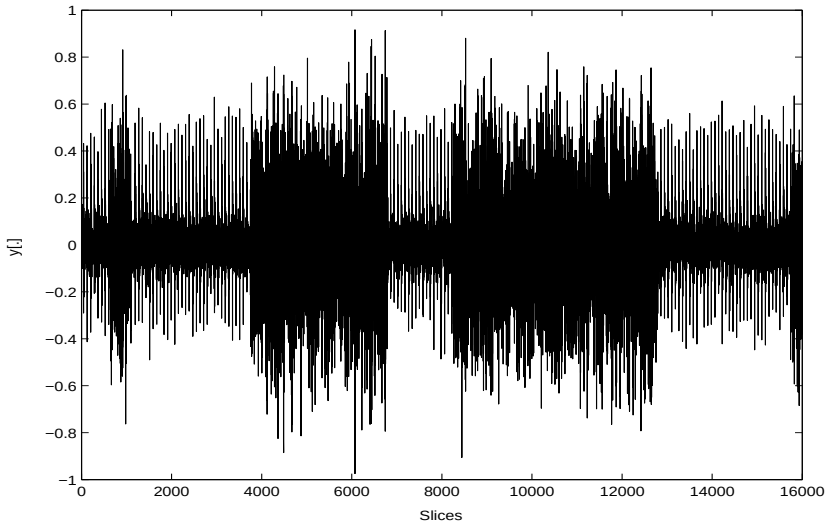


Abbildung 6.17: Mit $P_n = 0.01$ verrauschtes Eingangssignal.

Zweite Simulation, $P_n = 0.01$, Gain variabel

Erhöht man die Rauschleistung auf $P_n = 0.01$ zeigt sich das Problem, dass der Verstärkungsfaktor nicht mehr gefunden wird, er stellt sich zu hoch ein, daher sind die Amplituden der stimmhaften Abschnitte am Verstärkereingang (Abbildung 6.19) zu klein (vgl. Tabelle 3.2 und die Signalabschnitte werden nur mehr *'stimmlos'* bzw. *'stimmhaft low'* bewertet (Abbildung 6.20). Die Filterkoeffizienten in Abbildung 6.18 stellen sich erst gegen Ende unseres Signalausschnittes ein. Abbildung 6.21 zeigt die Ausgangsnachricht des Anregungsmodells. Die Mittelwerte der Nachricht stimmen zwar, die Unsicherheit der Schätzung sieht man in der höheren Varianz. Das Ausgangssignal selbst (Abbildung 6.22) zeigt, dass erst in dritten Stimmhaften Abschnitt eine kleine Verbesserung erzielt werden konnte.

Dritte Simulation, $P_n = 0.01$, fixer Gain

Wiederholt man obige Simulation mit der einzigen Änderung, dass man den Verstärkungsfaktor mit $\kappa = 1$ dem Graphen aufzwingt, wird das verrauschte Eingangssignal wieder richtig bewertet (Abbildung 6.24 und 6.25). Die Vorgabe $\kappa = 1$ wird durch den Apriori-Knoten erreicht, indem man in dessen Initialnachricht den Gain $g = 1$ mit minimaler Varianz $V = 10^{-6}$ setzt. Die Filtergewichte in Abbildung 6.23 stellen sich wieder schneller ein und das verrauschte Eingangssignal aus Abbildung (6.17 kann in diesem Fall sehr gut von der Störung befreit werden. Abbildung 6.26 zeigt das geschätzte unverrauschte Eingangssignal.

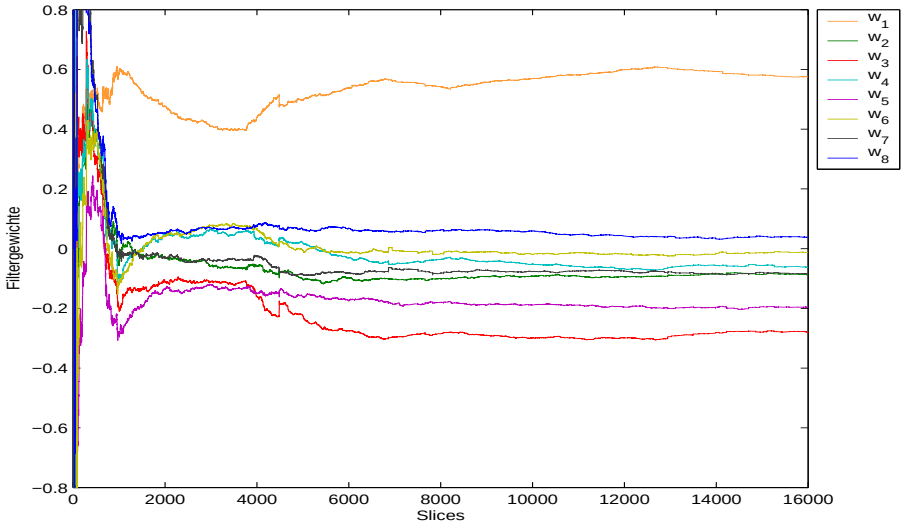


Abbildung 6.18: Die Filtergewichte stellen sich nur sehr langsam ein, da der Verstärkungsfaktor nicht richtig erkannt wurde.

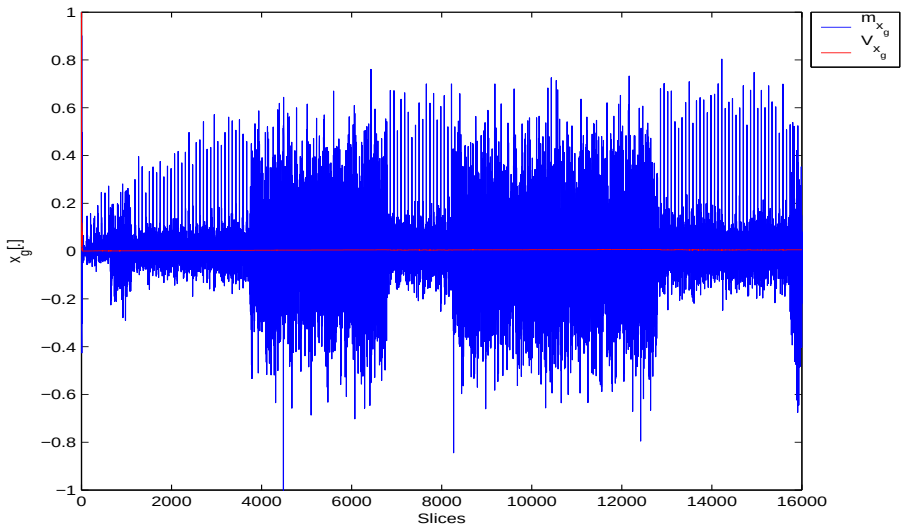


Abbildung 6.19: Mittelwert der Eingangnachricht des Verstärkungsknoten. Durch fehlerhaften Verstärkungsfaktor ist die Amplitude der Imulse zu gering.

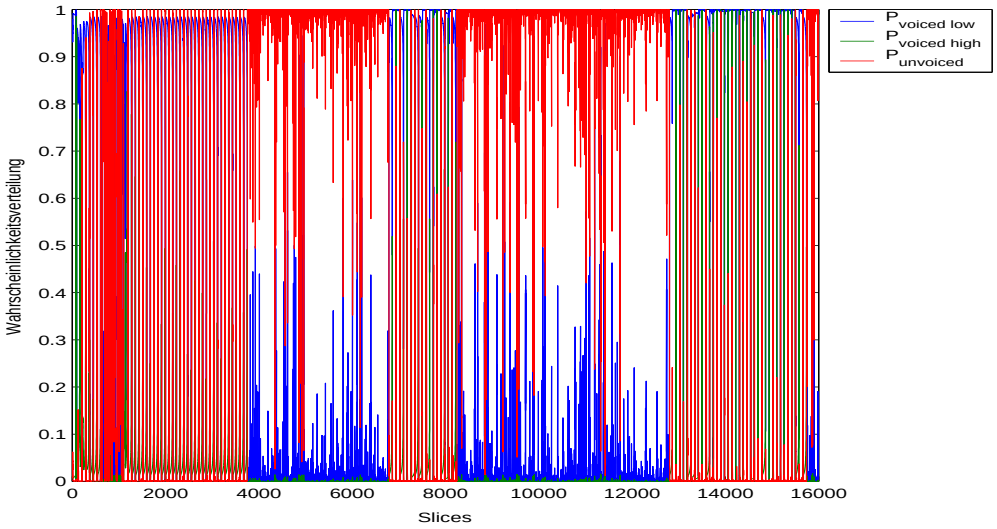


Abbildung 6.20: Der Steuerausgang der FSM zeigt ebenfalls, dass lange die richtigen Zustände nicht gefunden werden.

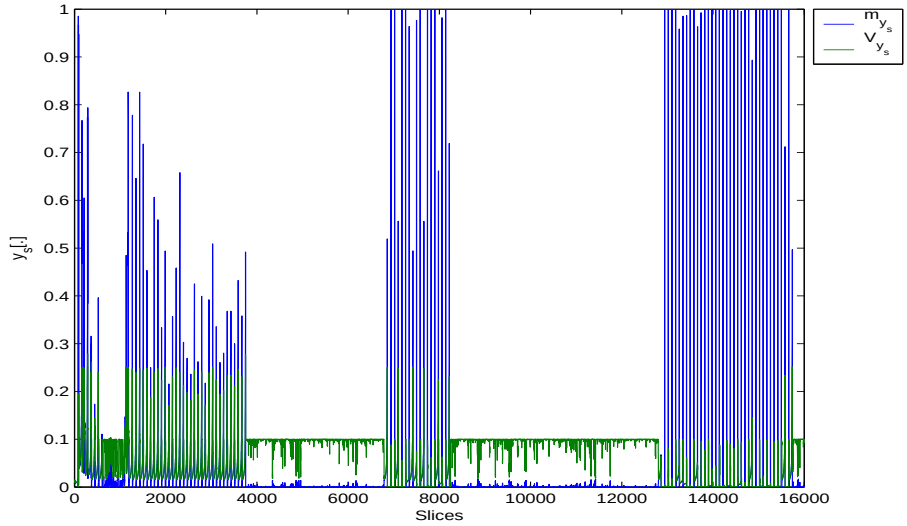


Abbildung 6.21: Mittelwert und Varianz des Anregungssignals am Schalterausgang sind im ersten Abschnitt sehr ungenau.

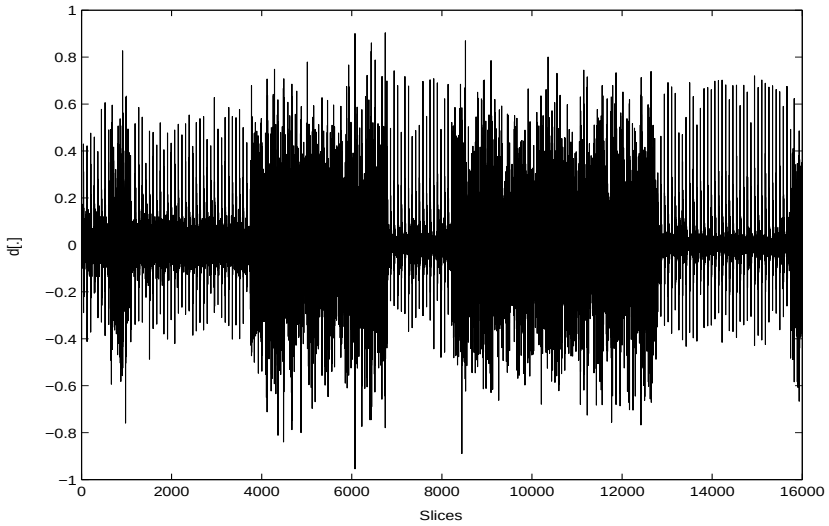


Abbildung 6.22: Das Rekonstruiertes Ausgangssignal zeigt erst im letzten stimmhaften Abschnitt eine leichte Signalverbesserung.

Vierte Simulation, $P_n = 0.1$, fixer Gain

Wenn die Rauschleistung um eine weitere Zehnerpotenz auf $P_n = 0.1$ erhöht wird, muss der Verstärkungsfaktor wieder festgelegt werden. Die Abbildungen 6.28 bis 6.31 zeigen wieder die verschiedenen Nachrichten im FFG. Vergleicht man das verrauschte Eingangssignal in Abbildung 6.27 und das gewonnene Ausgangssignal in Abbildung 6.32, so sieht man, dass die Störgeräuschunterdrückung prinzipiell sehr gut funktioniert. In Abbildung 6.30 sieht man, dass im stimmlosen Abschnitt die Wahrscheinlichkeitsverteilung nicht immer ganz eindeutig ist. Graund dafür ist, dass sich die Gaußverteilungen der beiden Zustände *'voiced low'* und *'unvoiced'* am Schalter bei höherer Varianz immer ähnlicher werden (vgl. Abschnitt 4.1, Abbildung 4.3).

6.3.3 Variable Filterkoeffizienten

In diesem Abschnitt werden nun auch die Filterkoeffizienten nach Abbildung 4.4 verändert. Zur Simulation verwenden wir das Signal, wie wir es in Abschnitt 6.2.1 bereits verwendet haben. Der Faktor-Graph soll unter Verwendung des Eingangssignals aus Abbildung 4.6 den Verlauf der Filterkoeffizienten (Abbildung 4.5) bei Anwendung des Summe-Produkt

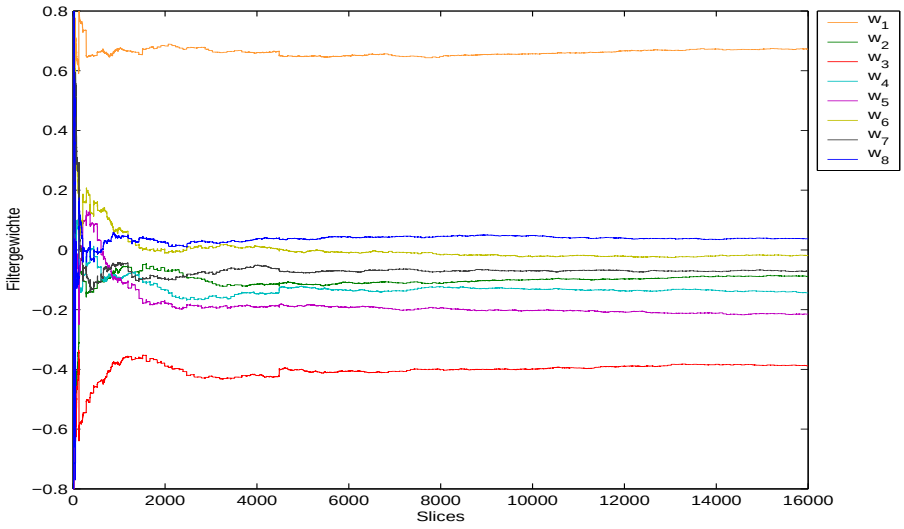


Abbildung 6.23: Die Filtergewichte stellen sich bei vorgegebenen Verstärkungsfaktor wieder schneller ein.

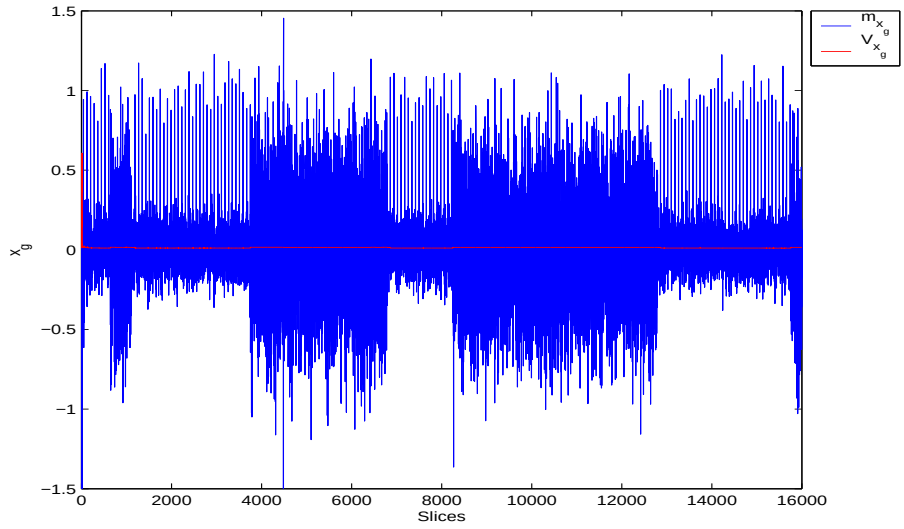


Abbildung 6.24: Mittelwert der Impulse am Schalterausgang ist von Beginn an groß genug um richtig bewertet zu werden.

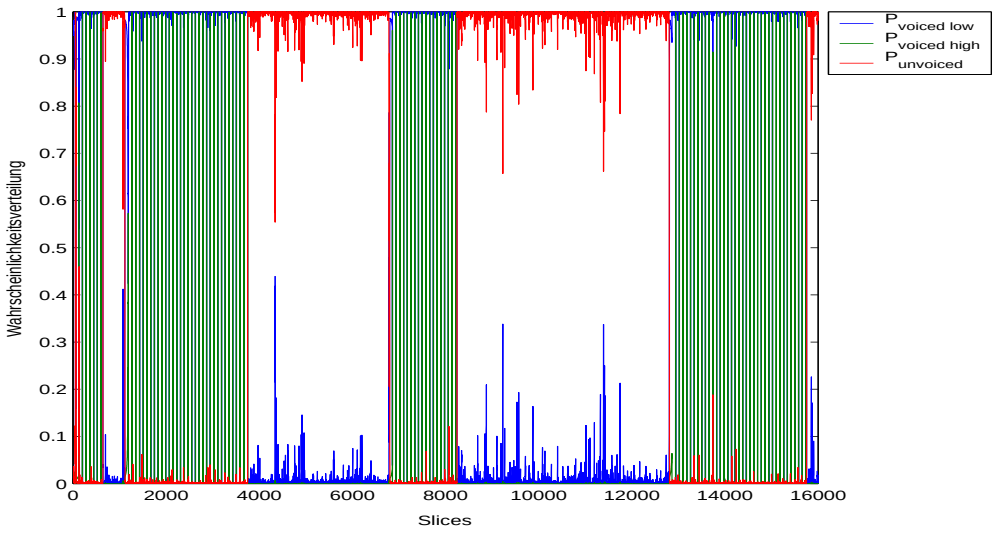


Abbildung 6.25: Die FSM stellt somit ebenfalls sofort die richtigen Zustände ein.

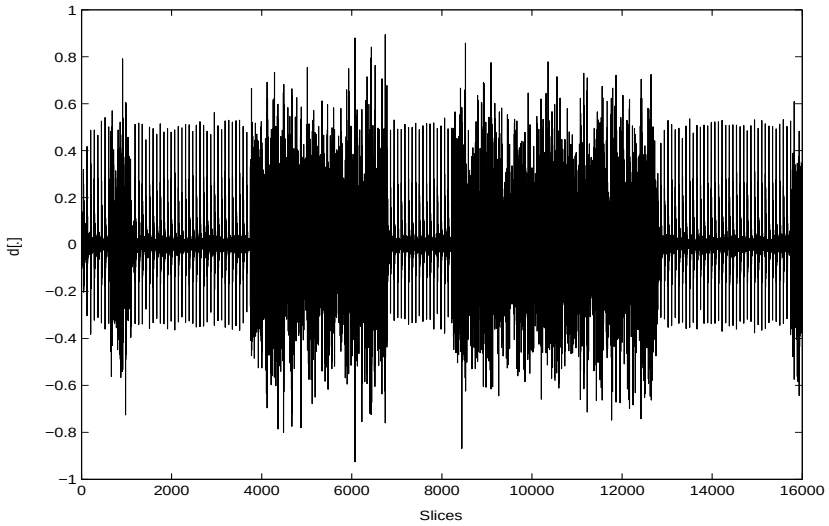


Abbildung 6.26: Wir erhalten eine sehr gute Schätzung des rauschfreien Signals, wobei $P_n = 0.01$ war.

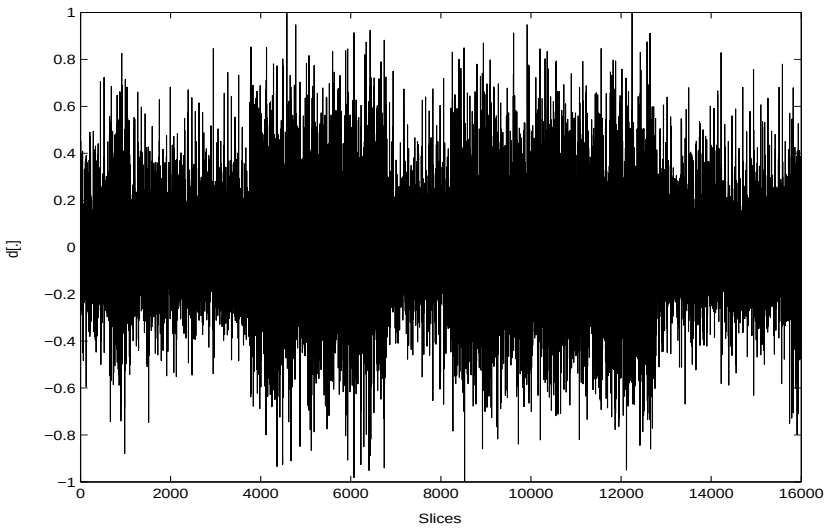


Abbildung 6.27: Mit $P_n = 0.1$ verrauschtes Eingangssignal

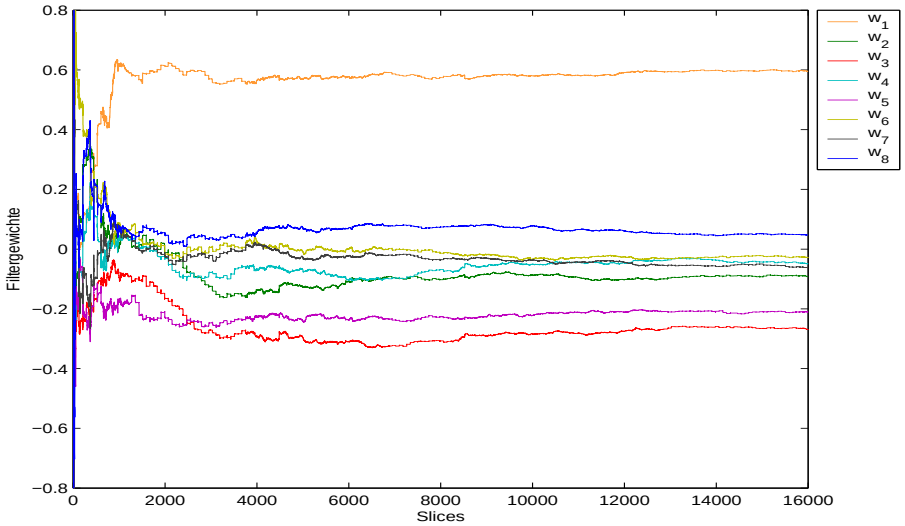


Abbildung 6.28: Je höher die Rauschleistung, umso unsicherer ist vor allem am Beginn die Berechnung der Filtergewichte.

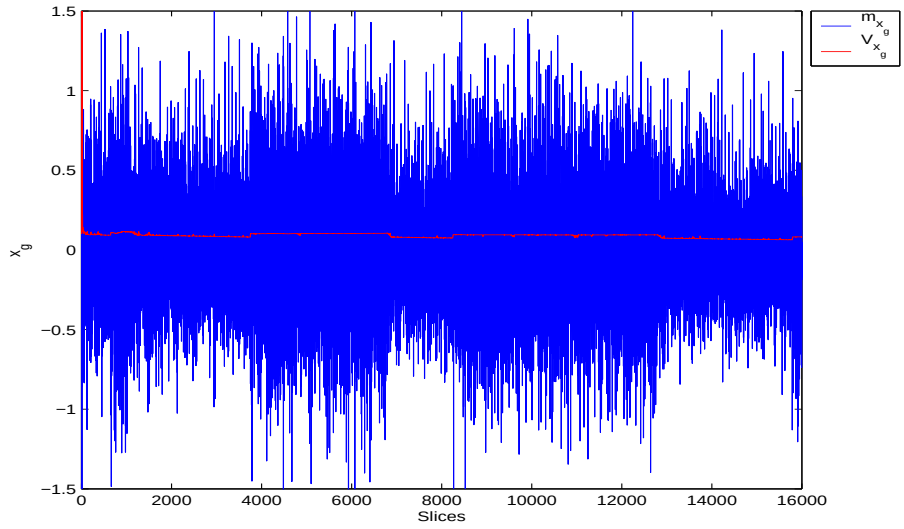


Abbildung 6.29: Die Nachricht am Schalterausgang ist ebenso stärker verrauscht.

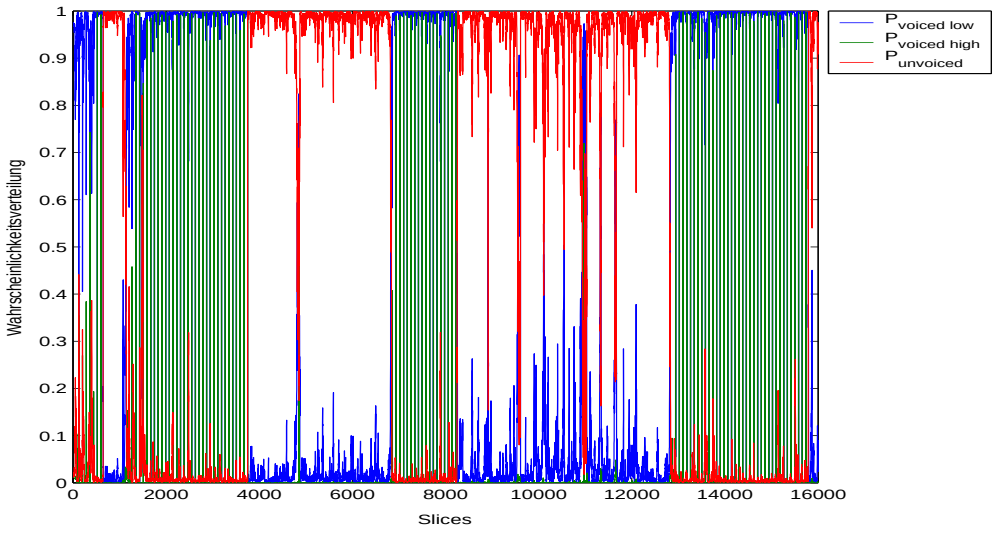


Abbildung 6.30: Wahrscheinlichkeitsverteilung am Steuerausgang der FSM ist in stimmhaften Bereichen immer noch eindeutig, im stimmlosen Abschnitt ist öfters eine Unsicherheit zwischen *'voiced low'* und *'unvoiced'* zu bemerken.

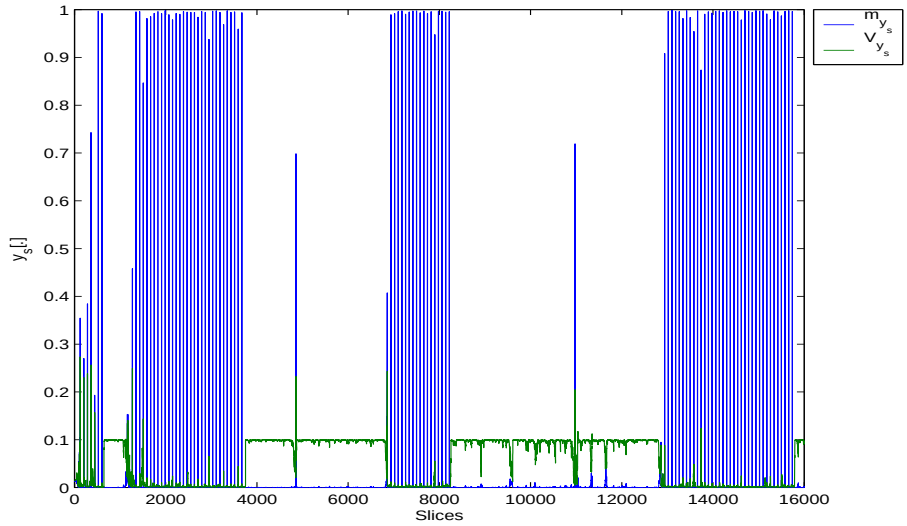


Abbildung 6.31: Mittelwert und Varianz des geschätzten Anregungssignals.

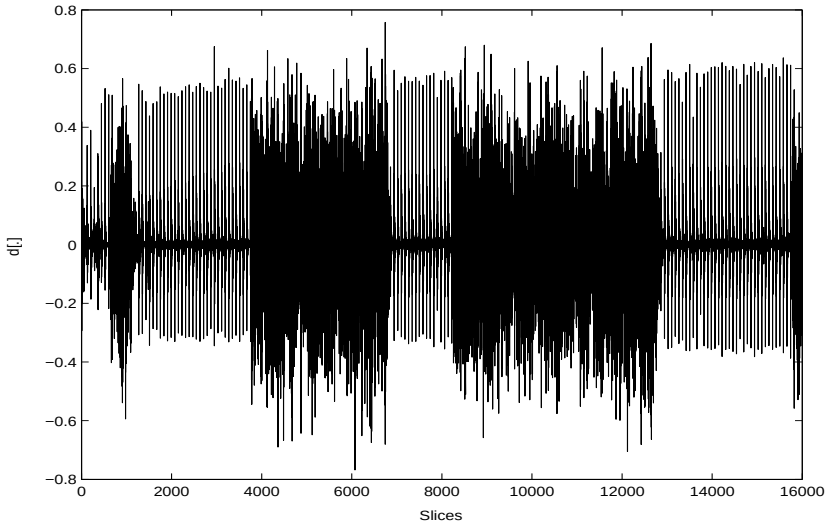


Abbildung 6.32: Vom Rauschen befreites Ausgangssignal wobei $P_n = 0.1$ war.

Algorithmus wiedergeben.

Erste Simulation, kein Störgeräusch

Abbildung 6.34 zeigt den zeitlichen Verlauf der errechneten Filterkoeffizienten unter Verwendung von Abbildung 6.33 als Eingangssequenz. Nach dem Einschwingvorgang zeigt sich, dass der errechnete Verlauf dem zuvor Simulierten sehr nahe kommt. In Abbildung 6.36 ist die Wahrscheinlichkeitsverteilung der drei Schalterzustände, wie sie von der FSM berechnet wurde, zu sehen. Es wird richtig zwischen stimmhaft und stimmlos unterschieden, in den ersten beiden stimmhaften Abschnitten ist diese Entscheidung jedoch noch nicht so eindeutig. In Abbildung 6.37 sieht man einen Ausschnit von 250 Slices im Detail. Der Mittelwert der Impulse ist noch nicht groß genug, um am Schalter beim Vergleich mit den Gaußverteilungen der drei Zustände (vgl. Abbildung 4.2) nur mehr in den Zustand *'voiced high'* zu fallen.

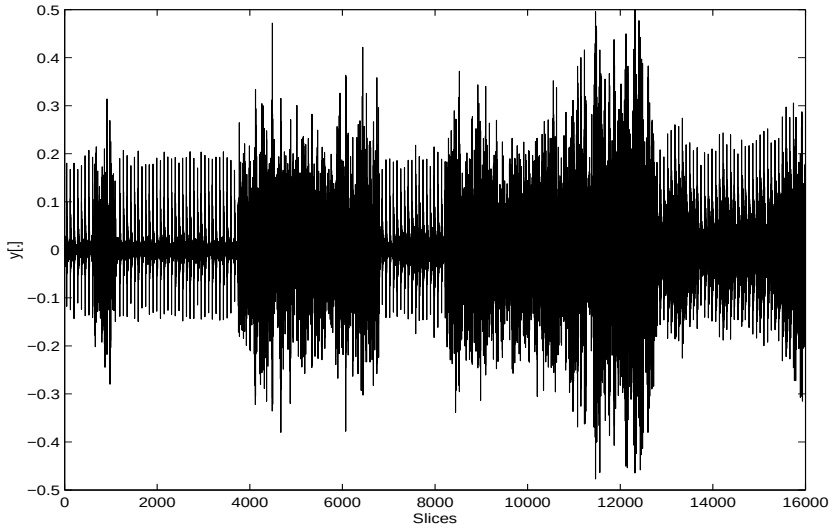


Abbildung 6.33: Eingangssignal zum testen des FFG ohne Rauschen

Zweite Simulation, mit Störgeräusch, fixer Gain

Wir generieren ein verrauschtes Signal mit $P_n = 0.01$ (Abbildung 6.39) mit dem Verlauf der Filterkoeffizienten aus Abbildung 4.5. Dieses Signal soll nun vom Rauschen befreit werden, den Gain $g = 1$ geben wir durch den Apriori-Knoten vor. Wie man in Abbildung 6.40 sieht, stellen sich die Filtergewichte wieder schnell ein. In den ersten 11000 Slices verlaufen

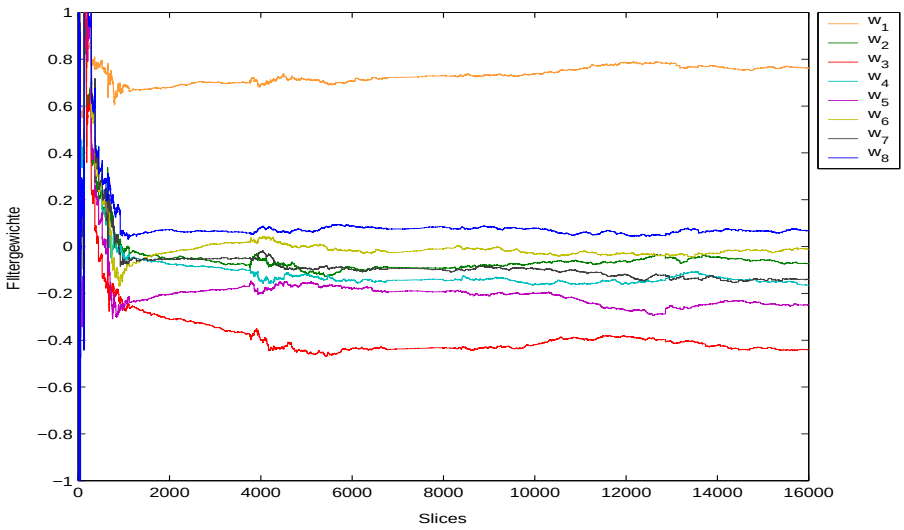


Abbildung 6.34: Verlauf der Filterkoeffizienten ohne Rauschen

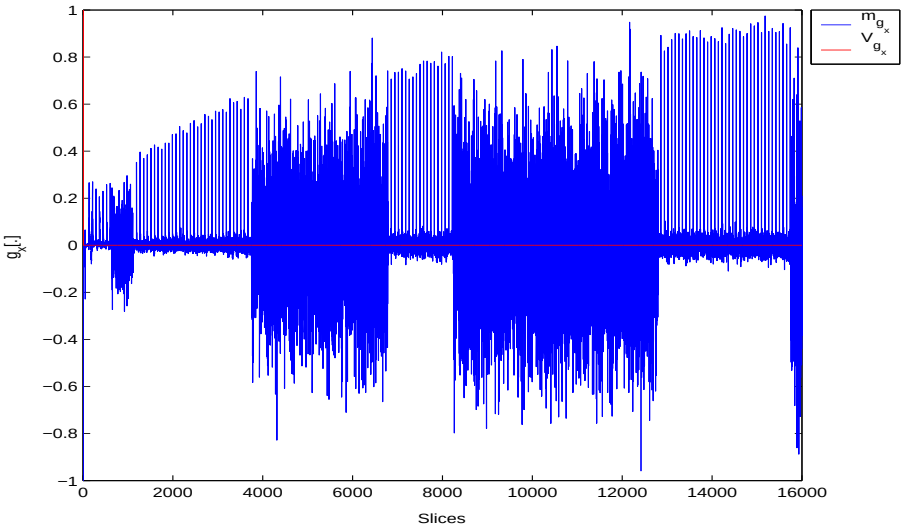


Abbildung 6.35: Die Amplitude des Impulszuges der Nachricht vom Verstärkungsknoten zum Schalter geht nur langsam Richtung Eins.

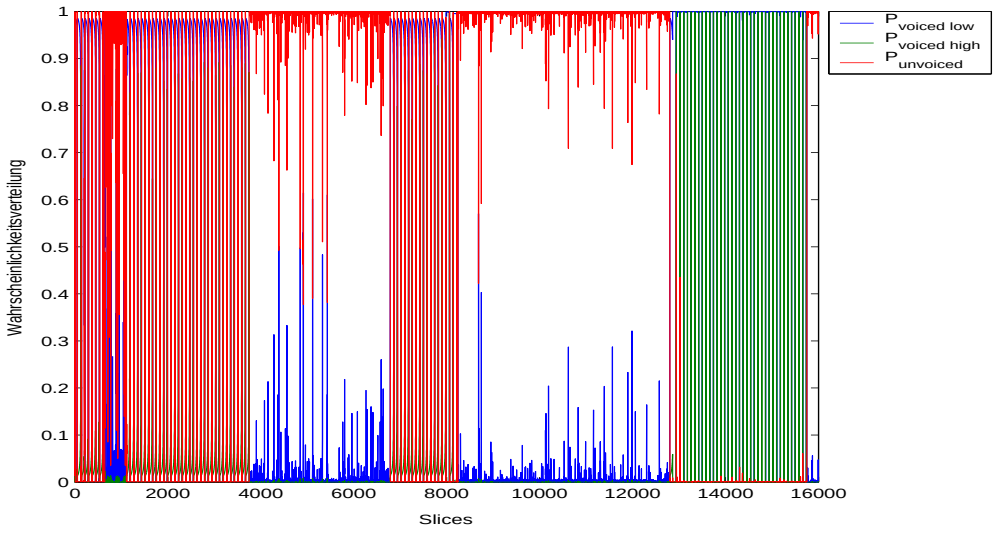


Abbildung 6.36: Ausgang der FSM ohne Rauschen

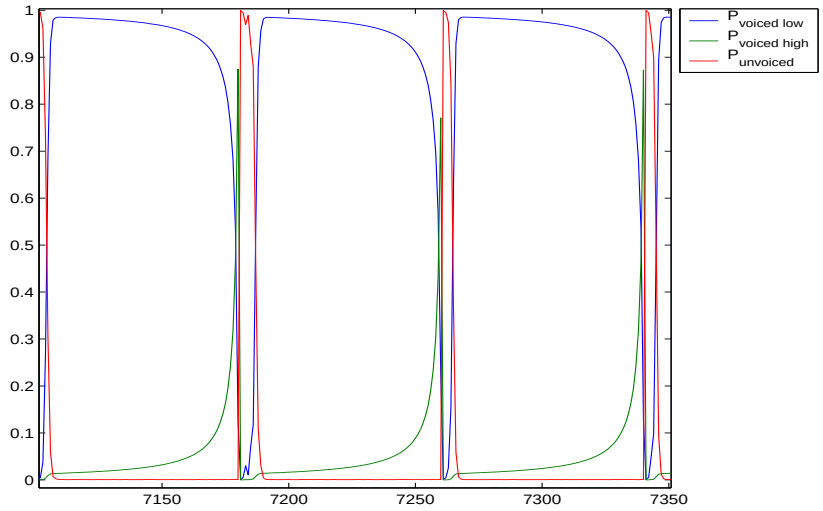


Abbildung 6.37: Detailansicht eines Ausschnitts des Signals aus Abbildung 6.36

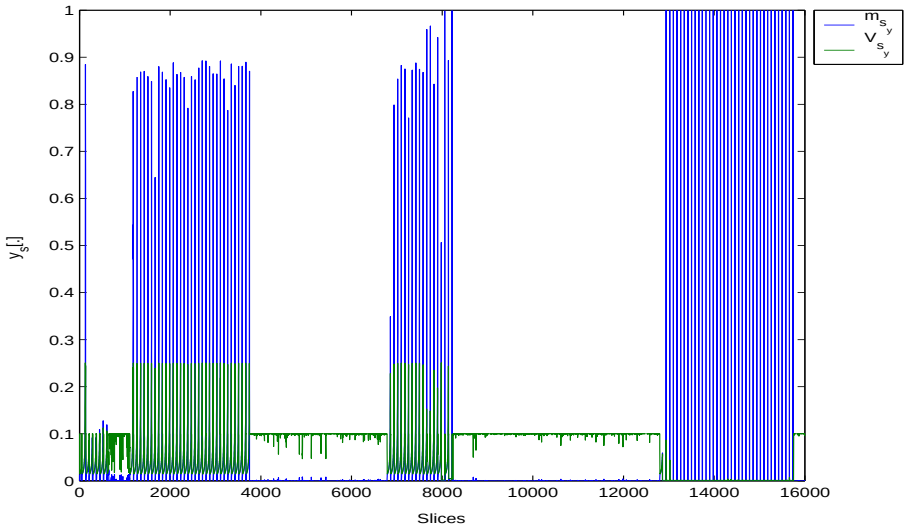


Abbildung 6.38: Auswirkungen der nicht eindeutigen Schalterstellung sieht man an der höheren Varianz bei den ersten beiden stimmhaften Abschnitten des Anregungssignals.

sie ähnlich unserer Erwartung. Auffällig sind dann die Bewegungen im Bereich von Slice 11000 bis 13000. Bei der Betrachtung der weiteren Nachrichten im Graphen sieht man, dass es genau in diesem Bereich einige *Ausreisser* in der Nachricht vom Verstärkungs- zum Schalterknoten gibt. Diese hohe Spitzen fallen dann natürlich wieder in keine der drei Gaußverteilungen unserer möglichen Zustände. Entstanden sind diese Peaks am Additionsknoten vom Filter mit dem Eingangssignal. Die Filterkoeffizienten beginnen wegzudriften, kehren aber nach dem Wechsel auf einen stimmhaften Signalabschnitt wieder zu ihren korrekten Werten zurück. Während diesem Abschnitt des Wegdriften der Filterkoeffizienten werden keine richtigen Schalterstellungen erkannt.

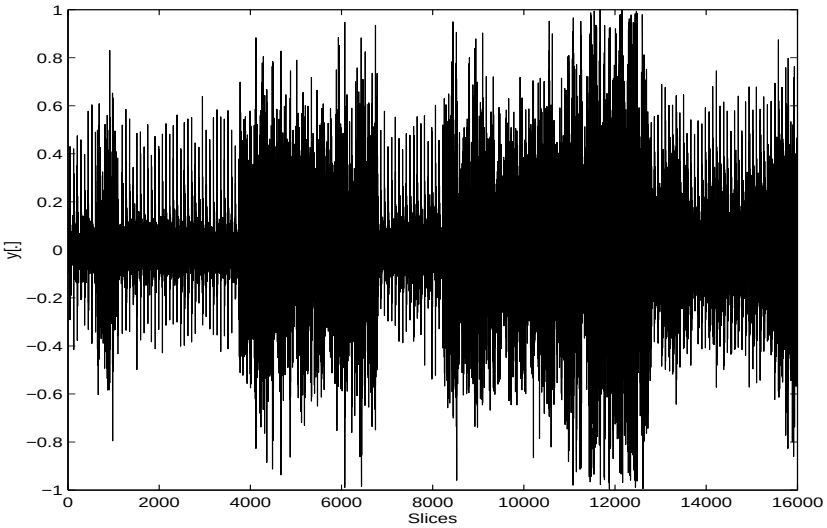


Abbildung 6.39: Eingangssignal zum testen des FFG mit Rauschen, $P_n = 0.01$

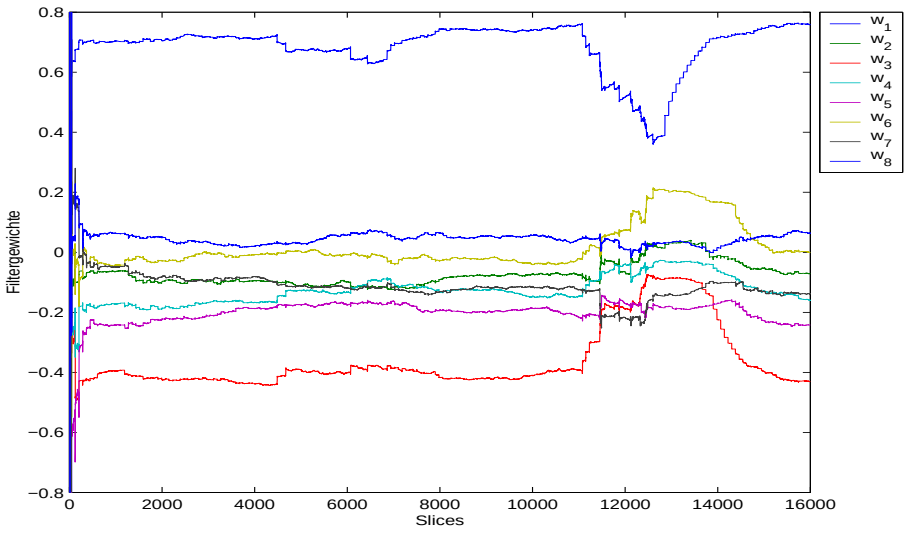


Abbildung 6.40: Zeitlicher Verlauf der Filterkoeffizienten

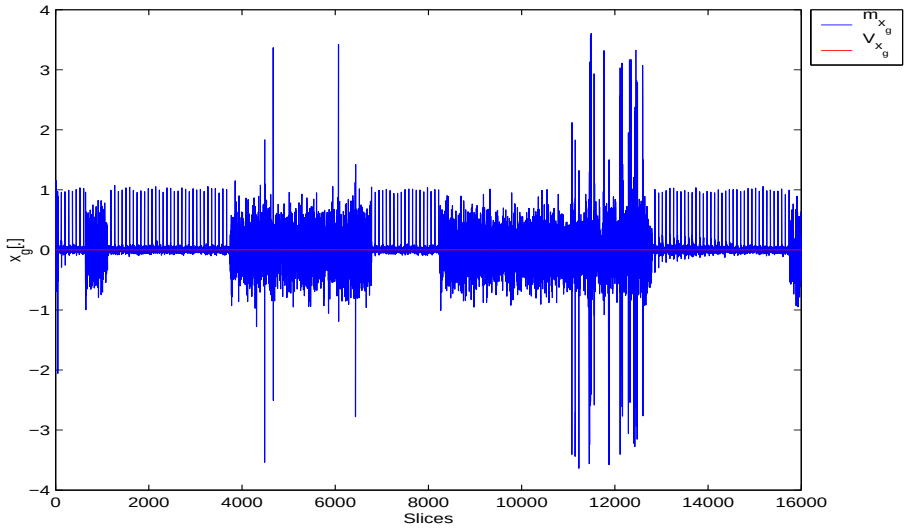


Abbildung 6.41: Die Amplitude des Mittelwertes der Nachricht vom Verstärkungsknoten zum Schalter wurde mit $\kappa = 1$ festgelegt.

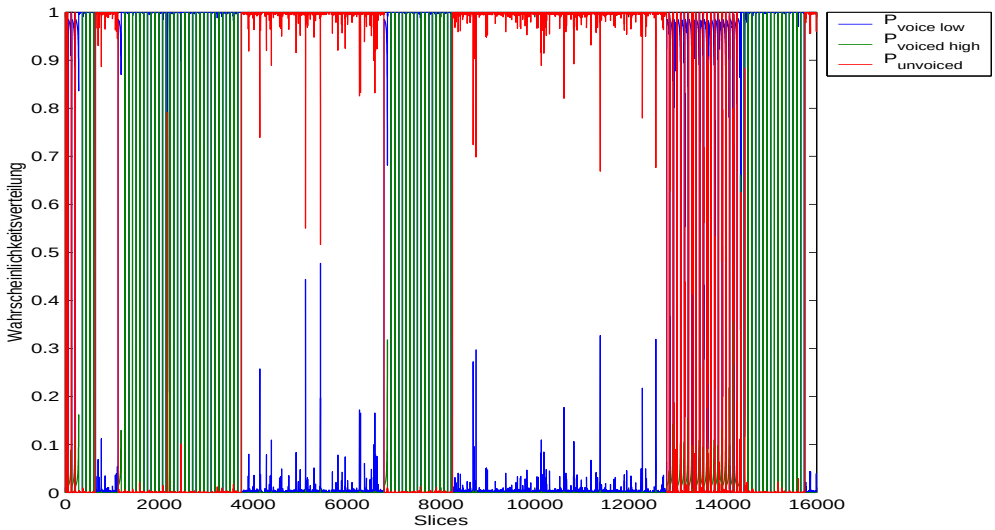


Abbildung 6.42: Ausgang der FSM. In dem Bereich der stark veränderten Filtergewichte werden die Impulse nicht erkannt.

Kapitel 7

Zusammenfassung und Ausblick

In diesem Kapitel sollen die aus dieser Arbeit gewonnenen Erkenntnisse und Resultate abschließend diskutiert und mögliche Erweiterungen des bestehenden Systems aufgezeigt werden.

7.1 Zusammenfassung

In dieser Arbeit wurde an einem Modell zur Sprachverarbeitung in der Faktor-Graphen Darstellung gearbeitet. Ausgehend vom LPC-Sprachmodell ist dieses Modell entstanden und wurde nun durch einige Modifikationen verbessert.
Im ersten Teil wurde das alte Modell untersucht und beschrieben, um so die Möglichkeiten des Ausbaus klar zu machen. Es wurde gezeigt, dass in der FFG-Darstellung ein Modell sehr einfach erweitert und ausgebaut werden kann. So konnte eine Nachbildung eines Störgeräusches eingebunden werden, um so die Leistungsfähigkeit zur Störgeräuschbefreiung zu testen. Im weitern Verlauf wurde die vorhandene Modellierung der Filterkoeffizienten verändert. Bislang wurden diese über die Zeit konstant gehalten, was einer nicht sehr guten Nachbildung der Wirklichkeit entspricht. Diese Einschränkung wurde aufgehoben und so ein *'random walk'* der Filterkoeffizienten über die Zeit ermöglicht. Diese Möglichkeit der kontinuierlichen Änderung der Koeffizienten erlaubt uns eine bessere Nachbildung realer Sprachsignale.
Der zweite Teil befasste sich weniger mit dem Modell selbst, sondern mehr mit dem Aufbau und den Berechnungen mit Faktor-Graphen. Bislang wurden nur FFG einer endlichen Länge aufgebaut und zur Berechnung verwendet. Da das Ziel aber in der Echtzeitverarbeitung liegt, wurde eine ganz neue Methode der Handhabung eines Graphen entwickelt. Es wird gezeigt, wie man einen Graphen aufbaut und über die Zeit erweitert bzw. alte Teile entfernt.

Dies ermöglicht uns eine dynamische Verarbeitung nicht nur von Sprachsignalen. Mit Hilfe dieser Methode kann man die Berechnungen sehr flexibel gestalten und auf verschiedene Anforderungen, die Rechenzeit betreffend, eingehen.
In einem weiteren Teil der Arbeit wurden die verschiedenen Faktor-Graphen getestet. Dazu wurden selbst generierte synthetische Signale verwendet und simuliert. Die Test- und Programmierumgebung wurde von MATLAB auf C++ verlegt, da mit dieser Hochsprache große Datensätze simuliert werden können. Die theoretischen Überlegungen aus Kapitel 4 und 5 konnten so anhand von Simulationen bestätigt werden. Überprüft wurden das Einschwingverhalten des Graphen, die Konvergenz der einzelnen Variablen und die Leistungsfähigkeit zur Signalrestauration. Durch den dynamischen Auf- und Abbau des FFG haben wir nun die Möglichkeit mit Signalen zu simulieren, welche reale Sprachsignale bereits besser abbilden. Zusammenfassend lassen sich fogende Aussagen tätigen:

- Das Einschwingverhalten der Größen im FFG ist im Allgemeinen sehr gut. Es genügen wenige Iterationen und die Größen konvergieren zu einem Fixwert. Der Schedule hat dabei großen Einfluss.
- Bei Hinzufügen des Störmodells am Eingang konvergieren die Filtergewichte nicht mehr, bewegen sich jedoch in einem akzepablen Rahmen.
- Weißes Gauß'sches Rauschen kann durch den FFG eliminiert werden.
- Über die Zeit variable Filterkoeffizienten werden durch die Modelländerung vom FFG wieder sehr gut nachgebildet.
- Der FFG kann auch dynamisch aufgebaut werden; so kann Rechenzeit gespart werden.
- Der dynamische Graph aus nur einem Slice hat ein längeres Einschwingverhalten. Es werden aber auch die richtigen Filterkoeffizienten gefunden.

Folgende Schwachstellen haben sich wieder bemerkbar gemacht:

- Die Approximation der Nachricht am Verstärkerknoten bewirkt, dass der Verstärkungsfaktor nicht immer gefunden wird.
- Die Modellbildung des Schalters funktioniert für die Nachricht am Kontrolleingang bei höheren Störgeräuschen nicht mehr gut.

7.2 Ausblick

Die weitere Entwicklung an diesem Modell lässt sich in die Arbeitsbereiche 'System - Erweiterung', 'Modell - Verbesserung' und 'Simulations - Umgebung' unterteilen.

System - Erweiterung

Wie die Abbildung 1.1 zeigt, teilt sich das gesamte System auf mehrere Modelle auf. Bis jetzt wurde am Sprachmodell gearbeitet und ein einfaches Störmodell eingebunden. Die Implementation eines Modells zur Wiedergabe von Raumeigenschaften steht noch aus. Ein allgemein gültigeres Störgeräuschmodell ist noch zu entwickeln.

Modell - Verbesserung

Wie bereits erwähnt bereitet derzeit die Approximation der Gain-Nachricht Probleme. Um die Verstärkung zeitvariabel zu gestalten, kann, wie bei den Filtergewichten, ein *'random walk'* eingebunden werden. Dazu muss man eventuell eine bessere Beschreibung der Gain-Nachricht entwickeln. Um stimmhafte Signalanteile besser nachzubilden, sollte sich die Frequenz des Impulszuges geringfügig ändern dürfen. Diese Modellverfeinerung kann zum Beispiel durch flexiblere Übergangswahrscheinlichkeiten in der Matrix der FSM erfolgen.

Simulations - Umgebung

Die Simulationsumgebung ist direkt in die C++ Programmierung eingebunden, wobei gewünschte Plots über GNUPLOT ausgegeben werden. Diese Simulationsumgebung lässt sich noch komfortabler gestalten.

Anhang A

Erklärungen zu den C++ Programmen

Die Vorarbeit [4] wurde in MATLAB entwickelt und programmiert. Um mit realen Audiosignalen zu arbeiten und Frames mit einigen Sekunden zu simulieren und zu berechnen, wurde die Entwicklungsebene auf C++ gewechselt. Basisklassen und Grundfunktionen wurden mir freundlicherweise von Sascha Korl zur Verfügung gestellt. Darauf aufbauend wurde von mir weiterentwickelt.

A.1 FFG Klassen

Für das Erstellen eines Faktor-Graphen sind die drei Basisklassen *ffg*, *node* und *message* vorhanden. Für den Datenaustausch (Plot, Ausgabe einzelner Werte, Speicherung) eines FFG gibt es die Klassen *wavfile*, *gnuplot* sowie *journal*. Hier eine kurze Beschreibung der Klassen und ihrer abgeleiteten Klassen.

A.1.1 Klasse *ffg*

name	Name des FFG
nodes{}	Liste aller Knoten des FFG
messages{}	Liste aller Messages des FFG
ffg()	Konstruktor
~ffg()	Dekonstruktor
addnode()	Fügt einen neuen Knoten in den FFG ein
removenode()	Entfernt Knoten
connect()	Verbindet zwei Knoten
disconnect()	Trennt eine Verbindung zwischen zwei Knoten
schedule_add()	Fügt Ausgang eines Knotens in den Schedule ein
check()	Überprüfung aller Verbindungen
simulate()	Simuliert FFG
sim()	Simuliert dynamischen FFG
compute()	Berechnet Summe-Produkt Algorithmus am Graphen
comp()	Berechnet Summe-Produkt Algorithmus am dynamischen FFG
probe()	Gibt die Message an einem Port aus
utot()	Gibt die Randdichte an einer Kante aus

Abgeleitete Klasse *subgraph*

Die Klasse *subgraph* wird von der Basisklasse *ffg* und *node* abgeleitet. So kann ein Teil eines FFG zu einem Subgraphen mit einem eigenen Schedule zusammengefasst und als Typ *node* im FFG weiterverwendet werden.

A.1.2 Klasse *node*

name	Name des Knotens
ports{}	Liste der Ports des Knotens
node()	Konstruktor
~node()	Dekonstruktor

replicate()	Erstellt identischen Knoten
getname()	Gibt den Namen des Knotens aus
setname()	Setzt den Namen des Knotens
get_portnr_by_name()	Ausgabe der zum Namen gehörenden Portnummer
create_messobj()	Erstellt Nachricht an einem Port
plugport()	Verbindet Nachricht mit einem Port
unplugport()	Trennt Nachricht vom Port

Abgeleitete Klassen

node_fsm	Finite State Machine
node_switch	Schalter-Knoten
node_apriori	Apriori Wahrscheinlichkeit
node_gain	Gain-Knoten
node_equ	Gleichheitsknoten
node_add	Additionsknoten
node_index	Index-Knoten
node_innprod	Bildet Innprodukt
node_ssm	State Space Model

A.1.3 Klasse *message*

node_from	Knotenname, von dem Nachricht kommt
node_to	Knotenname, zu dem Nachricht geht
portnr_from	dazugehörige Portnummer
portnr_to	dazugehörige Portnummer
message()	Konstruktor
~message()	Dekonstruktor
setconndata()	Verbindet Knoten und Port

Abgeleitete Klassen

mess_gauss	Gauß'sche Wahrscheinlichkeitsdichte
mess_gain	Wahrscheinlichkeitsdichte des Verstärkungsfaktors
mess_discrete	Diskrete Wahrscheinlichkeitsdichte

A.1.4 Weitere Klassen

Klasse *wavfile*

Mit Hilfe der Klasse *wavfile* können die Wahrscheinlichkeitsverteilungen an den Kanten als wav-File gespeichert werden, bzw. Audiosignale eingelesen und verarbeitet werden.

file	File-Handle
setup	Setup-Handle
wavfile()	Konstruktor
~wavfile()	Dekonstruktor
open()	File öffnen
close()	File schließen
writesample()	Sample in File schreiben
readsample()	Sample auslesen
numsample()	Gibt Frame Count zurück

Klasse *gnuplot*

Die Klasse *gnuplot* ermöglicht das direkte Zugreifen auf *Gnuplot* um die Nachrichten im Graphen auch graphisch darstellen zu können.

handle	Gnuplot Handle
titles	Liste mit Plot-Legende
gnuplot()	Konstruktor
~gnuplot()	Dekonstruktor
open()	Plot öffnen
reset()	Plot löschen
close()	Plot schließen
cmd()	Gnuplot-Command verwenden
setstyle()	Linientyp des Plots
addtitle()	Legende anzeigen
plotx()	Kurve zeichnen
plotdata()	Set von Kurven mit Legende zeichnen
savedata()	Speichert Plotdaten in File

Klasse *journal*

Die Klasse *journal* verwendet die Standardklasse *fstream* und beinhaltet die Ausgabe für Vektoren und Matrizen der verschiedenen Nachrichtentypen.

A.2 Erstellen eines FFG

In diesem Abschnitt sei anhand eines einfachen Beispiels erklärt, wie in der C++ Umgebung ein FFG aufgebaut und handgehabt wird. Abbildung A.1 zeigt nochmals das Störquellenmodell aus Kapitel 4. Im Folgenden wird beschrieben, wie dieses Modell implementiert wird.

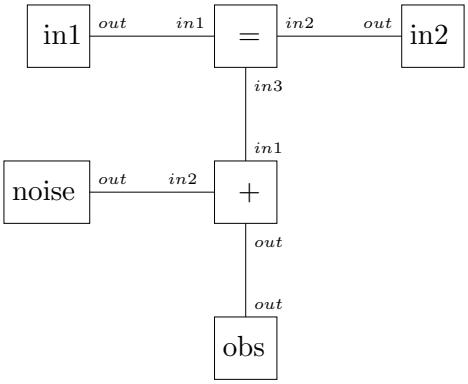


Abbildung A.1: FFG des Störsignalmodells mit den Bezeichnungen der Ports

Als erstes wird ein Element der Klasse *ffg* erzeugt. *'g'* ist dann der Faktor-Graph mit dem Namen *'noiseinput'*. Als Eingangssignal wird ein gesamtes Audiosignal eingelesen. Die Daten werden dann im Falle eines dynamischen Graphen mitübergeben.

```
ffg g("noiseinput",verbose);
read_vector(obs,"input.wav",rwv_wav,SAMPLERATE);
```

Der Aufbau des FFG erfolgt mit der Funktion *'addnode()'*, die Randknoten sind jeweils Apriori-Knoten mit Standard-Initialisierung, oder, wie beim Knoten *'obs'*, mit dem eingelesenen Audiosample initialisiert.

```
g.addnode(new node_apriori<mess_gauss>("obs",new
mess_gauss(obs[0],MINV),
verbose));
```

```
g.addnode(new node_apriori<mess_gauss>("in1",new mess_gauss(1),verbose));
g.addnode(new node_apriori<mess_gauss>("in2",new
mess_gauss(1),verbose));
```

Der Gleichheits-, der Additions- und der Apriori-Knoten *'noise'* werden zu einem Subgraphen zusammengefügt mit dem Namen *'slice'*.

```
psl[i]=new subgraph("slice",verbose);
psl[i]->addnode(new node_apriori<mess_gauss>("noise",new
mess_gauss(0,NOISEVAR),verbose));
psl[i]->addnode(new node_add("addnoise",1,verbose));
psl[i]->addnode(new node_equ<mess_gauss>("equin",1,verbose));
```

Im Folgenden werden die Knoten des Subgraphen verbunden und die Ports nach außen mit neuen Namen versehen. Mit *'addnode()'* wird der Subgraph als Knoten dem FFG *'g'* hinzugefügt.

```
psl[i]->connect("equin","in3","addnoise","in1");
psl[i]->connect("noise","out","addnoise","in2");
psl[i]->defineport("inleft","equin","in1");
psl[i]->defineport("inright","equin","in2");
psl[i]->defineport("obs","addnoise","out");
g.addnode(psl[i]);
```

Die Ports des Subgraphen müssen noch mit den Apriori-Knoten verbunden werden. Abbildung A.2 zeigt den entstanden FFG.

```
g.connect("slice","inleft","in1","out");
g.connect("slice","inright","in2","out");
g.connect("slice","obs","obs","out");
g.check();
```

In diesem Fall interessieren uns die Nachrichten an den Ports *'inleft'* und *'inright'* des Subgraphen. Für den Schedule genügt es also, die Nachrichten vom Eingang her in Richtung der beiden Ports mit dem Summe-Produkt Algorithmus zu berechnen. Der Schedule würde dann folgendermaßen lauten:

```
g.schedule_add(true,"obs","out");
g.schedule_add(true,"slice","noise","out");
g.schedule_add(true,"slice","addnoise","in1");
```

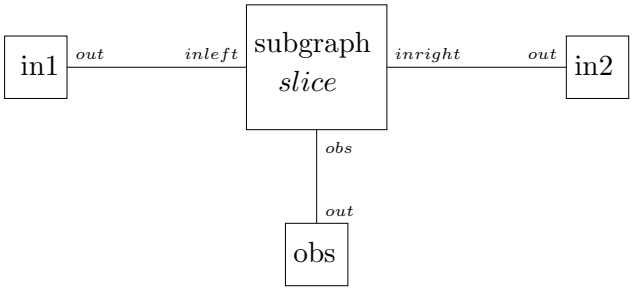


Abbildung A.2: FFG mit Subgraph und den Portbezeichnungen

```
g.schedule_add(true,"slice","equin","in1");
g.schedule_add(true,"slice","equin","in2");
```

Gestartet wird die Berechnung mit

```
g.compute();
```

Um sich die Randverteilung nun ausgeben zu lassen, muss zuerst der Wert an einer Stelle mit der Funktion *'probe()'* geholt werden und kann dann einerseits mit der *'jrn'* Funktion ausgegeben oder mit *'gnuplot'* ein Plot erstellt werden.

```
pmg=dynamic_cast<mess_gauss*>(psl[i]->probe("equin","in1"));
data[0][i]=pmg->mean[0];
jrn<<data[0][i];
pmg=dynamic_cast<mess_gauss*>(psl[i]->probe("equin","in2"));
data[1][i]=pmg->mean[0];
jrn<<data[1][i];
plot.open();
plot.setstyle("lines");
plot.cmd("set grid");
plot.cmd("set xlabel 'iteration'");
plot.cmd("set ylabel 'coeff'");
plot.cmd("set yrange [-1:1]");
s.str("");s<<"set xrange [0:"<<NITERS-1<<"]";
plot.cmd(s.str());
plot.plotdata(data,"in1.mean","in2.mean");
cin.get();
plot.close();
```

Literaturverzeichnis

[1] H.-A. Loeliger. *Digitale Signalverarbeitung 2*. Vorlesungsskript ETH Zürich, Sommersemester 2001.

[2] Frank R. Kschischang, Brendan J. Frey, Hans-Andrea Loeliger. *Factor Graphs and the Sum-Product Algorithm*. IEEE Trans. Information Theory, vol. 47, no 2, pp. 498-519, Februar 2001

[3] Hans-Andrea Loeliger. *Least Squares and Kalman Filtering on Forney Graphs*,in *Codes, Graphs, and Systems*,(R.E.Blahut and R. Koetter, eds.), Kluwer Academic Publishers, 2002.

[4] Michael Koller. *Binaurale Ansätze zur Störgeräuschbefreiung in Hörgeräten*. Diplomarbeit, Institut für Signal- und Informationsverarbeitung,ETH Zürich, März 2002.

[5] Beat Pfister, Hans-Peter Hutter. *Sprachverarbeitung 1*. Vorlesungsskript ETH Zürich, 2001.