The Netcharge Method[1]

Applying Artificial Neural Networks to classify the b quark production flavor in the DELPHI experiment at CERN's Large Electron Positron accelerator.

Bernt Olav Rostad

High Energy Particle Physics Group
Institute of Physics
University of Oslo
Norway

August 27, 1996

[1]Thesis submitted to the degree of Cand.Scient at the University of Oslo

Abstract

This thesis outlines an alternative to the well known Jetcharge method, thus christened **The Netcharge Method**, for use in B_s mixing analysises. The aim of the method is to increase the efficiency of classifying the production flavor, or sign of the charge, of b quarks in $Z^0 \rightarrow b\bar{b}$ decays. The core of the Netcharge method is an Artificial Neural Network. Various net-structures and input variables are trained and tested. Results better than the Jetcharge method are obtained even though the parameter choices and net-structures can be optimized further.

Title : The Netcharge Method – Applying Artificial Neural Networks to classify the b quark production flavor in the DELPHI experiment at CERN's Large Electron Positron accelerator.

Author : Bernt Olav Rostad

Date : August 27 1996

Supervisor : Prof. Torleiv Buran

Keywords : B-tagging, B-mixing, B-oscillations, Jetcharge, Artificial Neural Networks

Publisher : University of Oslo
Institute of Physics
Box 1048 Blindern
N-0316 Oslo
Norway

Document : Written in LaTeX using the *report* documentclass with 12 points width.
All figures in encapsulated postscript, created by PAW and the Unix graphics tool xfig

Acknowledgements

This thesis was written in a very fertile environment at the HEP department of the University of Oslo. I'm grateful for all the comments and help I've received from my fellow students; Esben and Vidar Lund, Håkon Fløystad, Jørgen Hansen and Yngve Kvinnsland. Fredrik Glöckner's hints about LaTeX has also been appreciated. I'm also grateful for the useful on- and off-topic discussions with prof. Alex Read and prof. Lars Bugge during numerous morning coffee-meetings. Graduate student Trond Myklebust has been very helpful when soft- and hardware problems occured, which happened almost daily. I would also like to express a large gratitude to my supervisor, prof. Torleiv Buran, for sending me to several CERN conferences and the CERN Summer Lectures 1995, to pick up inspiration, and for his great patience with the progress of my work. The final outcome of this thesis would not have been possible without the practical and theoretical guidance from graduate student Ole Myren Røhne, last semester interactively from CERN. Thanks for two wonderful years!

I would like to dedicate this thesis to my family for its warm support and firm economical backup all these years.

Physics must not be mistaken for reality,
it's merely a good way of describing nature.

– Bernt Rostad, Blindern, August 1996

Contents

Chapter 1

Introduction

An introduction to CERN and the DELPHI detector is given, along with a brief description of the contents of this thesis.

1.1 CERN

In 1949 the french physicist Louis de Broglie proposed the creation of a European science laboratory. The aim was to unite European nuclear physics, which was in a bad state after the destructive war. On September 29 1954 the *Conseil Européen pour la Recherche Nucléaire*, or *European Organization for Nuclear Research*, was formally established with 11 member states. The CERN laboratory complex was located in Meyrin, a quiet countryside a few kilometers outside Geneva, not far from the french border (figure 1.1).

Today CERN focus more at particle physics than nuclear research, this is reflected by the change of name to *European Laboratory for Particle Physics*. There are currently 19 member states and CERN host several thousand users, roughly 50% of the worlds particle physicists, many from non-member states like the USA, India, Japan and Russia. Thus it is suitable to label CERN as the first worldscale collaborating particle physics laboratory.

In addition to particle physics CERN is the birthplace of the much hyped World Wide Web, originally developed in 1990-91.

1.2 The accelerator complex

This section will briefly list some details about the accelerator complex at CERN. But first a quick look at why so much energy is needed to detect those tiny particles.

1.2.1 Particles as probes

The energy in particle physics experiments today are usually several orders of magnitude larger than just a few decades ago. The reason for this is closely related to the wave-particle duality of Quantum Mechanics (QM). According to QM all particles can be assigned a wave property [1], with a wavelength given by $\lambda = \frac{h}{p}$, where h is the Planck constant. Just as photons are used to probe structures in optical microscopes one can use the particles to probe matter. But to be able to probe a given structure the wavelength must be shorter than the size of the structure.

For smaller structures the wavelength of the particle must be lowered, which amounts to increasing its momentum. Modern experiments probe the quark structure, at scales down to 10^{-17}m, a quick calculation gives:

$$E = h \cdot \frac{c}{\lambda} = 6.63{\cdot}10^{-34} Js \cdot \frac{3.00{\cdot}10^{8} ms^{-1}}{10^{-17} m} = 1.99 \cdot 10^{-9} J \approx 100 GeV$$

Thus modern accelerators need to accelerate particles up to 100 GeV in order to probe the inner structure of matter.

This may look like a perfect way of probing matter, if one wants to investigate even smaller structures just turn up the voltage to increase the energy of the probing particles. But there is one serious catch: During acceleration particles radiate photons and thereby lose energy.

1.2.2 Synchrotron radiation

In linear accelerators energy loss due to acceleration is rather small, because the actual acceleration only takes place in certain intervals. However, in circular accelerators the particles are accelerated most of the time, partly to increase their momenta but mainly to deflect their paths so that they travel around the ring. It is the bending of the particles that causes the synchrotron radiation, a kind of bremsestrahlung, which is the main source of energy loss at CERN's Large Electron Positron collider. The energy loss, ΔE, due to synchrotron radiation is proportinal to the energy of the particle [2]

$$\Delta E \propto \left(\frac{E}{m_0}\right)^4 \cdot \frac{1}{\rho}$$

Here m_0 is the rest mass of the particle, E its energy and ρ the bending radius of the accelerator. From this formula it is clear that an increase in the energy by a factor 2 would increase the energy loss by a factor 16. Equivalently lighter particles will lose much more energy from synchrotron radiation than heavier, in fact an electron will lose a factor $(\frac{m_p}{m_e})^4 \approx (1840)^4 \approx 10^{13}$ more than a proton with the same energy.

To reduce the energy loss at high energy circular accelerators one have two possibilities. Either one can use heavy particles, like protons, which is

the solution chosen for the future Large Hadron Collider (LHC) at CERN, where protons will be accelerated to 7 TeV. The other way of reducing the energy loss is by using a very large bending radius, since ΔE is inversly proportional to ρ. This was applied for the construction of LEP.

Figure 1.1: CERN is located near the point where the small and the large circle, indicating the SPS and LEP rings, coincides. The dotted line is the border between France and Switzerland, Geneva airport is seen near the bottom. The large dots on the LEP ring represents detectors, the dot closest to the airport indicates the position of DELPHI.

1.2.3 SPS and LEP

At CERN the first really large circular accelerator was the Super Proton Synchrotron (SPS), built in the 1970's, which could accelerate protons up to 400 GeV. To be able to do so it needed a circumference of 6 km. It was too large to fit inside the Swiss border, but the french government allowed CERN to build the SPS under french territory. The famous UA1 experiment, which made the first observations of the W and Z bosons in 1983, was in connection with the SPS.

But there are some drawbacks with hadron colliders, linked to the fact that hadrons are compound particles. The main problem is that the total momentum of the particle is shared by its constituent quarks. In a collision usually only two of the quarks will interact, the rest are spectators. Thus only a small fraction of the total energy is available to create new particles. Secondly, a hadron collision will result in more noise. The debris of particles

created by the spectator quarks will fill up the detector with unwanted tracks. For these reasons CERN decided to build a large electron positron collider.

From 1983 to 1988 the Large Electron Positron (LEP) collider was the largest civil-engineering undertaking in Europe. The total circumference of the ring is 26.6589 km, by far the largest in the world. It is located between 80 and 170 meters below the ground and reach from the Geneva airport, in Switzerland, to the Jura mountains, in France. Along the tunnel there are eight caves, or pits, four of them contain the current LEP experiments: ALEPH, DELPHI, L3 and OPAL.

From its start-up in August 1989 until the fall 1995 LEP accelerated electrons and positrons to a total center of mass energy of 91.2 GeV, this era is known as LEP1. The energy was not chosen at random, it is exactly what is needed to create a real Z^0 boson. During the six years of LEP1 each of the experiments at LEP collected millions of Z^0 decays, from which very precise measurements of electroweak parameters could be made. One of the most important results from LEP1 is that there are only three light neutrino generations [3] so, unless there is a neutrino with mass above 45 GeV/c^2, there are only three generations of quarks and leptons (section 2.1).

Other important results include precise measurements of the the running coupling constants, the weak mixing angle ($\sin^2\theta_w$), lepton universality and the hadronic branching fractions [4]. And as a curiosity the mass of the top quark was predicted to be 177 GeV, though with rather huge errors, from the LEP measurements [5]. This is in good agreement with direct measurements made at Fermilab.

In October 1995 the LEP energy was ramped up to 130 GeV as the first step in the LEP2 era. At LEP2 the energies will be increased in steps up to probably 195 GeV in 1998. The main goals at LEP2 are to look for the Higgs boson and possibly light supersymmetric particles, or to exlude such particles from the mass region covered by LEP2. When the LEP era ends in 2000-1 the LEP tunnel will be reused for the LHC project scheduled to start running in 2004.

1.3 The DELPHI detector

DELPHI, a DEtector with Lepton, Photon and Hadron Identification, is designed as a general purpose detector with special emphasis on powerful particle identification over a 4π solid angle. This means that DELPHI is built around the interaction point (IP), where the e^+ and e^- beams from LEP collide, with different subdetectors placed in layers radially outwards from the IP. Fully assembled DELPHI is three stories high and located 80 meters below the surface, in pit 8, along the LEP tunnel. Figure 1.2 should indicate the dimensions.

The analogy with an onion is often used when describing modern particle

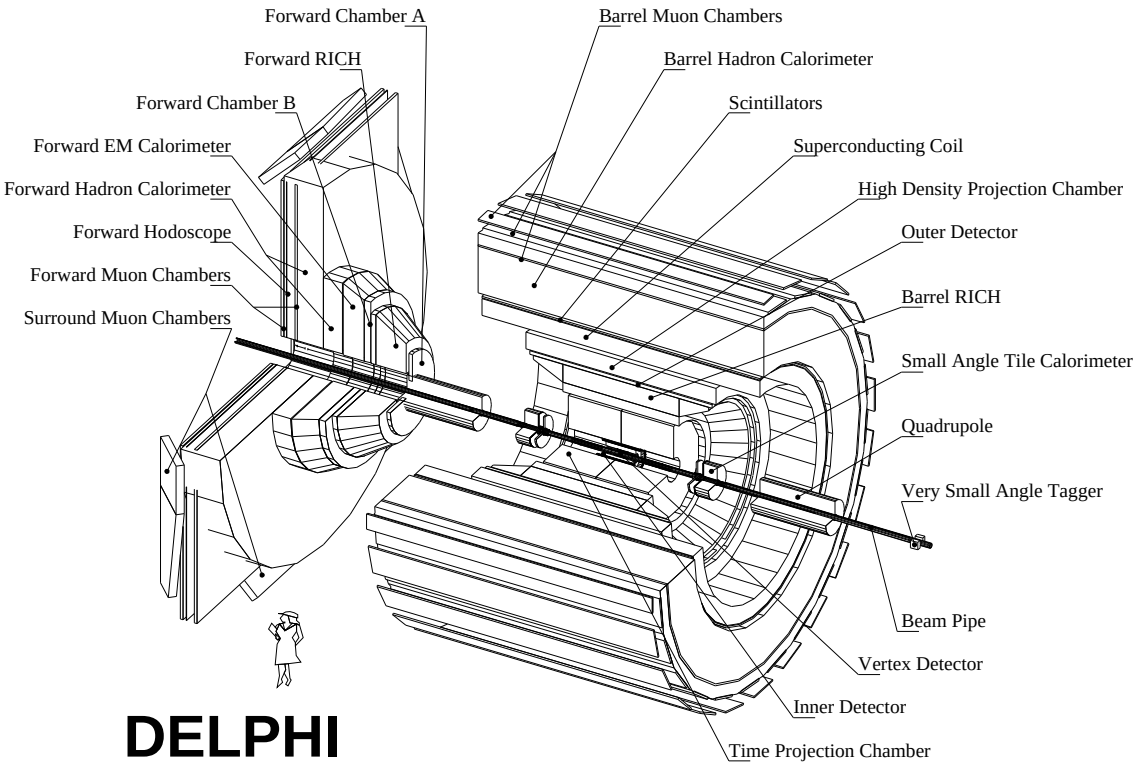


Figure 1.2: Cut-through view of the DELPHI detector

detectors, and DELPHI is no exception. The subdetectors of DELPHI are located in layers radially outwards from the beampipe which pass through the center of DELPHI. Each of the subdetectors has a unique task. Based on the information from one or several of them DELPHI can classify many different particles to a high degree of certainty. In DELPHI the z-axis is parallell to the beam, while the radius R and azimuth angle ϕ are in the plane perpendicular to the beam and θ is the polar angle, with θ=0 along z.

The main DELPHI subdetectors, from the center and outwards [6]:

Vertex Detector (VD) : Consists of 3 coaxial cylindrical layers of silicon strip detectors, with average radii of 6.3, 9.0 and 10.9 cm. The VD provides both $R\phi$ and Rz information and is used for impact parameter measurements and precise vertex reconstruction.

Inner Detector (ID) : Consists of an inner drift chamber with jet-chamber geometry providing $R\phi$ information. Surrounding the drift chamber are 5 cylindrical layers of multi-wire proportional chambers, providing Rz

information. The ID is used for vertex reconstruction and fast trigger information.

Time Projection Chamber (TPC) : Is a drift chamber where both endplates are divided in 6 azimuthal sectors, each sector containing sense wires and pads for more precise $R\phi$ determination. The z information is computed from the drift time along the chamber. The TPC is the main tracking device in DELPHI.

Ring Imaging CHerenkov detector (RICH) : Consists of a liquid and a gaseous radiator where particles can produce Cherenkov light cones. The light cones are reflected in mirrors and focused onto a photosensitive time projection chamber where photoelectrons are created and detected as ring images. This technology was very new at the time DELPHI was proposed but has turned out successful. The RICH can separate kaons, pions and protons from 1 GeV and up to 25 GeV.

Outer Detector (OD) : Consists of 5 layers of drift tubes located between radii 197 and 206 cm. The OD provides both $R\phi$ and Rz information. It is essential for fast trigger information and to improve the momentum resolution.

High density Projection Chamber (HPC) : Consists of 144 modules arranged in 6 rings. Each HPC module is a small TPC with layers of high density material in the gas volume. In this volume high energy electrons will create showers and the electron energy can be computed, with a precision of a few percent, based on the shape of the showers.

Hadron Calorimeter (HCAL) : Is installed in the return yoke of the DELPHI magnet and consists of 19000 streamer tubes. Hadrons will interact strongly with the dense material of the return yoke and create showers that will be detected in the streamer tubes. Electrons will not penetrate this far and muons will give very faint signals since they have no strong interaction, so the showers will be from hadrons. The energy resolution is of the order 25%

MUon Chambers (MUC) : Because muons do not interact strongly and are 200 times more massive than electrons they pass through most of the subdetectors with almost no energy loss. In addition the iron of the HCAL provides a filter for muons because the bulk of hadrons are stopped by this material. Hence the MUC is located as the outermost subdetector of DELPHI. The efficiency of detecting muons in MUC are between 80 and 90% with a misidentification of a few percent.

There are similar subdetectors in the forward and backward directions of DELPHI to cover most of the θ region. To measure the momentum of tracks,

in the TPC and OD, a highly uniform 1.2 T magnetic field is provided by a superconducting solenoid, with a 5000 A circulating current. The solenoid is located outside the HPC and cooled to 4.5 K by a high pressure flow of liquid helium. In addition to these particle identification detectors DELPHI has two subdetectors dedicated to measure the beam energy and luminosity:

Small angle TIle calorimeter (STIC) : Is a sampling lead-scintillator calorimeter. It is formed by two cylindrical detectors located along the beampipe, on both sides of the interaction point, in a distance of 2.2 meters. In 1994 STIC replaced SAT (Small Angle Tile calorimeter) as the main monitor for energy and luminosity in DELPHI. STIC gives an energy resolution of 2.7% at LEP1 beam energy.

Very Small Angle Tagger (VSAT) : Is located close to the beampipe, but 7.7 meters away from the interaction point, inside the LEP tunnel. It is used for relative luminosity measurements and provides fast beam background information.

The LEP bunch-crossing interval is 11 μs, which means that each second 90 thousand collisions occur in the center of DELPHI. In order to pick out the interesting events, without missing too many of the subsequent collisions, the trigger chain in DELPHI must work very fast. If an event is interesting the next one will be skipped to give the software time to read out the information, if not the event will be erased and the next one treated by the triggers when it arrives.

In summary DELPHI is a general purpose detector with a high efficiency of collecting interesting events. At the end of LEP1 it detected thousands of Z^0 decays each day of running and all in all several million Z^0 decays have been collected by DELPHI, providing material for very precise tests of the electroweak theory.

1.4 Concerning the thesis

This thesis was written using natural units, that is units where the speed of light and $\hbar$ is set to one, $c{=}\hbar{=}1$. Natural units are very popular in the HEP community because they simplify many formulas and give different physical quantities the same dimension.

Chapter 2 contains the basic theory behind B mixing while chapter 3 gives a brief summary of two experimental methods, used particularly in B physics, and some measurements of B mixing. In chapter 4 the basic theory behind neural networks is briefly discussed, with emphasis on feed-forward networks. In chapter 5 some important quantities, which will be used throughout the analysis, are defined. In chapters 6 and 7 the two parts of the analysis are outlined and results for different netstructures given. In chapter 8 the analysis is summarized and the conclusions drawn.

Chapter 2

Physical formalism

This chapter outlines the basic physical formalism used in this thesis, i.e. the concept of mixing, sometimes refered to as oscillations, in addition to a brief look at the standard model to clearify the notion of mesons.

2.1 Mesons

The standard model [7] of particle physics lists 6 quarks and 6 leptons, each with a corresponding anti-particle, as the fundamental building blocks of nature. Figure 2.1 shows the three known generations of quarks and leptons, the anti-particles are left out.

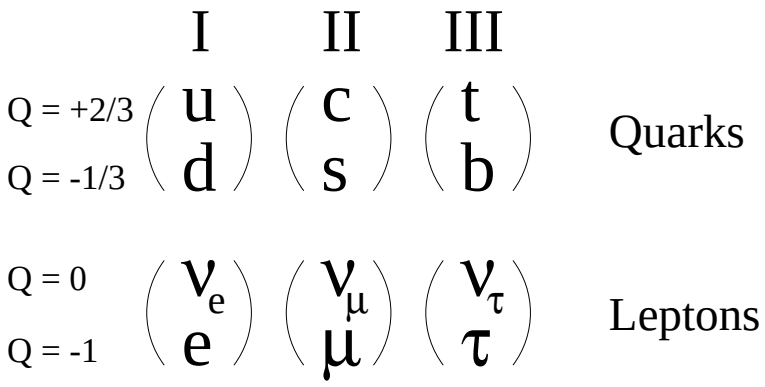


Figure 2.1: The three generations of elementary particles in the standard model. The charge is given in units of the elementary charge, the neutrinoes (ν) being neutral.

The anti-particles are equal to the particles in the sense that they have the same mass and follow the same laws of nature. The difference is that their quantum numbers (like charge, flavor, etc.) are reversed. An anti-particle is

usually denoted with a bar above the particle name, thus the anti-b quark is labeled $\bar{b}$.

According to the standard model, or more precisely Quantum Chromo Dynamics [8], all quarks carry color. There are three different colors, R, G and B. This has nothing to do with the macroscopic property color, but is a name given to an intrinsic property of quarks. The important thing is that nature appears to be colorless, thus free quarks can not be observed. The requirement of colorless states force nature to group the quarks in two possible ways [9].

1. Three quarks with different colors: $q_R + q_G + q_B$
2. One quark and one anti-quark with color and anti-color: $q_c + \bar{q}_{\bar{c}}$

Here $c \in [R, G, B]$. Particles from the first group are known as *baryons*, while particles from the second group are known as *mesons*. Protons and neutrons of atomic physics are examples of baryons, while pions and kaons are examples of mesons. This thesis will only be concerned with mesons, especially mesons containing a b or a $\bar{b}$ quark, known as B mesons:

$$B_q = \begin{pmatrix} \bar{b} \\ q \end{pmatrix} \qquad \bar{B}_q = \begin{pmatrix} \bar{q} \\ b \end{pmatrix} \tag{2.1}$$

In section 2.3 a phenomenon known as B mixing will be described, it can only occur among neutral B mesons due to conservation of the electric charge. According to definition (2.1) the q quark must have the same absolute value of its charge as the b quark to form a neutral B meson. From figure 2.1 it is clear that only the s and d quarks, with charge $-e/3$, qualify for this. Thus the only neutral B mesons one can form are B_d^0 and B_s^0, with corresponding anti-particles. With the given definition it is the B_q^0 meson that contains the $\bar{b}$ quark, while $\bar{B}_q^0$ contains the b quark.

In chapter 3 a brief summary of some special features and measurements concerning B mesons will be discussed.

2.2 Time evolution of an unstable particle

To obtain quantities that can be measured in experiments one have to apply Quantum Mechanics. In order to do so the problem must be defined in the proper formalism, which amounts to finding the Hamilton operator $\mathcal{H}$ of the system. Then one must solve the Schrödinger equation

$$i\frac{\partial}{\partial t}|\Psi\rangle = \mathcal{H}|\Psi\rangle \tag{2.2}$$

where $|\Psi\rangle$ is a wave function describing the time evolution of the system. For an unstable particle the hamiltonian is $\mathcal{H} = (\mathcal{M} - i?/2)$, where $\mathcal{M}$ is the

mass operator and ? represents the decay width. The solution of Eq (2.2), using this expression for $\mathcal{H}$, is

$$i\frac{\partial}{\partial t}|\Psi\rangle = (\mathcal{M} - i?/2)|\Psi\rangle$$

$$\Downarrow$$

$$\int \frac{\partial|\Psi\rangle}{|\Psi\rangle} = -i(\mathcal{M} - i?/2)\int \partial t$$

$$\Downarrow$$

$$\ln|\Psi(t)\rangle = \ln|\Psi_0\rangle - i(\mathcal{M} - i?/2)\cdot t \tag{2.3}$$

In the integration it was used that $\mathcal{H}$ is time independent and thus can be placed outside the integrand. The time integration was performed from time t=0 to t, with the initial condition $|\Psi(\text{t=0})\rangle = |\Psi_0\rangle$. By exponentiating Eq (2.3) one can find an expression for the time evolution of $|\Psi\rangle$

$$|\Psi(t)\rangle = e^{-i(\mathcal{M}-i\Gamma/2)\cdot t}\cdot|\Psi_0\rangle \tag{2.4}$$

This gives $|\Psi(t)|^2 \propto e^{-\Gamma t} = e^{-t/\tau}$, hence it makes sense to define $\tau \equiv \frac{1}{\Gamma}$ as the lifetime of the unstable particle.

2.3 B mixing

After the discovery of the b quark scientists started to look for mixing among neutral B mesons, in analogy with mixing among neutral kaons [10]. This section briefly outlines the basic formalism of B mixing. Experiments and results related to B mixing are mentioned in section 3.4.

In this section $|B^0\rangle$ will denote the B_q^0 state and $|\bar{B}^0\rangle$ the $\bar{B}_q^0$ state, the resulting formalism will be the same whether q=d or q=s. These two states are CP conjugate, which means that by performing a combined charge and parity transformation (CP) on one of the states it should turn into the other: $\text{CP}|B^0\rangle = \eta_1|\bar{B}^0\rangle$ and $\text{CP}|\bar{B}^0\rangle = \eta_2|B^0\rangle$, where η_1 and η_2 are arbitrary phase factors usually set to -1. Neutral B mesons are split in two mass eigenstates, a light and a heavy, which can be written as linear combinations of the CP conjugate states [11]:

$$|B_L\rangle = q\cdot|B^0\rangle + p\cdot|\bar{B}^0\rangle \qquad |B_H\rangle = q\cdot|B^0\rangle - p\cdot|\bar{B}^0\rangle \tag{2.5}$$

2.3.1 Mixing formulas

In the Standard Model mixing occur via second order weak interactions (the strong interaction conserves the b flavor and thus can not explain mixing). There are two possible box diagrams, shown in figure 2.2. Since the quark

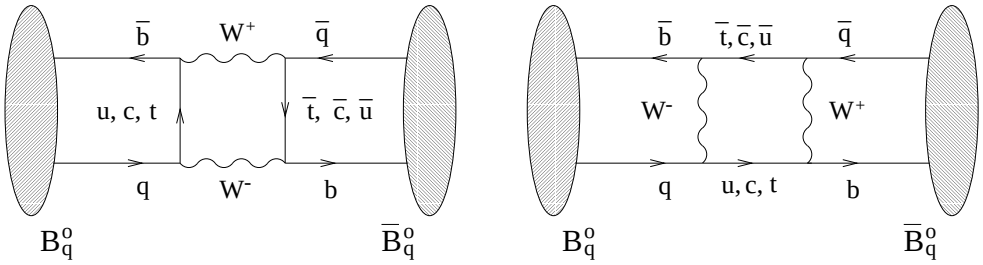


Figure 2.2: Second order box diagrams for B mixing in the Standard Model

couplings are proportional to the square of the mass it is obvious that the heaviest quark, the top, give the most dominant contribution. Thus indirectly the size of the mixing can reveal much about the mass of the top.

For the $B^0\bar{B}^0$ system, where $B^0 \rightleftharpoons \bar{B}^0$ transitions can occur, the total wave function can be written

$$|\Psi\rangle = \begin{pmatrix} |B^0\rangle \\ |\bar{B}^0\rangle \end{pmatrix} \tag{2.6}$$

The hamiltonian $\mathcal{H}$ must be non-Hermitian to allow for particle decay, if $\mathcal{H} = \mathcal{H}^\dagger$ the number of particles would have been conserved through unitarity. It is also clear that $\mathcal{H}_{11} = \mathcal{H}_{22}$ because of CPT invariance, and that $\mathcal{H}_{12} \neq 0$ and $\mathcal{H}_{21} \neq 0$ due to non-conservation of the quark flavor (to allow for B mixing). Finally the imaginary parts of $\mathcal{M}_{12}$ and $?_{12}$ must be nonzero to allow for CP violation. In summary [12]

$$\mathcal{H} = \begin{pmatrix} \mathcal{M} - i?/2 & \mathcal{M}_{12} - i?_{12}/2 \\ \mathcal{M}^*_{12} - i?^*_{12}/2 & \mathcal{M} - i?/2 \end{pmatrix}$$

But the indirect CP violation, due to different widths of the $|B_L\rangle$ and $|B_H\rangle$ mass eigenstates, are expected to be small. Thus to a good approximation the imaginary parts of $\mathcal{M}_{12}$ and $?_{12}$ can be set to zero, making the hamiltonian symmetric along both diagonals. The resulting Schrödinger equation is

$$i\frac{\partial}{\partial t}|\Psi(t)\rangle = \begin{pmatrix} \mathcal{M} - i?/2 & \mathcal{M}_{12} - i?_{12}/2 \\ \mathcal{M}_{12} - i?_{12}/2 & \mathcal{M} - i?/2 \end{pmatrix} \cdot |\Psi(t)\rangle \tag{2.7}$$

where $|\Psi(t)\rangle$ is given by Eq (2.6). Without indirect CP violation the ratio $|\frac{p}{q}|$ equals 1 [11], which modifies the mass eigenstates of Eq (2.5) to

$$|B_L\rangle = \frac{1}{\sqrt{2}} \cdot (|B^0\rangle + |\bar{B}^0\rangle) \qquad |B_H\rangle = \frac{1}{\sqrt{2}} \cdot (|B^0\rangle - |\bar{B}^0\rangle) \tag{2.8}$$

These are also eigenstates for the hamiltonian in Eq (2.7). This can be shown by expressing $\mathcal{H}$ in its ($|B^0\rangle$, $|\bar{B}^0\rangle$) basis:

$$\mathcal{H} = \mathcal{H}_{11}|B^0\rangle\langle B^0| + \mathcal{H}_{22}|\bar{B}^0\rangle\langle\bar{B}^0| + \mathcal{H}_{21}|B^0\rangle\langle\bar{B}^0| + \mathcal{H}_{12}|\bar{B}^0\rangle\langle B^0|$$

$$= H_1 \cdot (|B^0\rangle\langle B^0| + |\bar{B}^0\rangle\langle \bar{B}^0|) + H_2 \cdot (|B^0\rangle\langle \bar{B}^0| + |\bar{B}^0\rangle\langle B^0|)$$

Where $H_1 = \mathcal{M} - i?/2$ and $H_2 = \mathcal{M}_{12} - i?_{12}/2$ has been used. By using this expression for $\mathcal{H}$ it is straight forward to show that

$$\mathcal{H}|B_L\rangle = (H_1 + H_2)|B_L\rangle = \epsilon_L|B_L\rangle$$

$$\mathcal{H}|B_H\rangle = (H_1 - H_2)|B_H\rangle = \epsilon_H|B_H\rangle$$

where ϵ_L and ϵ_H are the eigenvalues of $\mathcal{H}$ with respect to the eigenstates $|B_L\rangle$ and $|B_H\rangle$. It can be shown that $\epsilon_L = M_L - i?_L/2$ and $\epsilon_H = M_H - i?_H/2$.

Let the initial state $|\psi(\text{t=0})\rangle = |B^0\rangle$ which from Eq (2.8) can be expressed as $|B^0\rangle = \frac{1}{\sqrt{2}}(|B_L\rangle + |B_H\rangle)$. Then in the same fashion as Eqs (2.2) - (2.4) one can solve Eq (2.7)

$$|\psi(t)\rangle = e^{-i\mathcal{H}t} \cdot |B^0\rangle = \frac{1}{\sqrt{2}} e^{-i\mathcal{H}t} \cdot (|B_L\rangle + |B_H\rangle)$$

By using that $|B_L\rangle$ and $|B_H\rangle$ are eigenstates of $\mathcal{H}$ the next step follows naturally.

$$|\psi(t)\rangle = \frac{1}{\sqrt{2}} \cdot \left(|B_L\rangle e^{-i\epsilon_L t} + |B_H\rangle e^{-i\epsilon_H t}\right)$$

$$= \frac{1}{\sqrt{2}} \cdot \left(|B_L\rangle e^{-i(M_L - i\Gamma_L/2)t} + |B_H\rangle e^{-i(M_H - i\Gamma_H/2)t}\right)$$

The final step is to use Eq (2.8) to obtain an expression in $|B^0\rangle$ and $|\bar{B}^0\rangle$

$$|\psi(t)\rangle = \frac{1}{2}\left(e^{-i(M_L - i\Gamma_L/2)t} + e^{-i(M_H - i\Gamma_H/2)t}\right)|B^0\rangle$$

$$+ \frac{1}{2}\left(e^{-i(M_L - i\Gamma_L/2)t} - e^{-i(M_H - i\Gamma_H/2)t}\right)|\bar{B}^0\rangle$$

To clearify this result it is useful to define

$$A_{B\to B}(t) = \frac{1}{2}\left(e^{-i(M_L - i\frac{\Gamma_L}{2})t} + e^{-i(M_H - i\frac{\Gamma_H}{2})t}\right) \tag{2.9}$$

$$A_{B\to \bar{B}}(t) = \frac{1}{2}\left(e^{-i(M_L - i\frac{\Gamma_L}{2})t} - e^{-i(M_H - i\frac{\Gamma_H}{2})t}\right) \tag{2.10}$$

so that the solution can be written as

$$|\psi(t)\rangle = A_{B\to B}(t)|B^0\rangle + A_{B\to \bar{B}}(t)|\bar{B}^0\rangle \tag{2.11}$$

According to the standard interpretation of Quantum Mechanics, the sixty year old *Copenhagen Interpretation*, the abstract wave function $|\Psi(t)\rangle$ is just the probability amplitude of finding a particle with the given quantum numbers at the given position and time. From Eq (2.11) one can verify that

$A_{B\to B}(t) = \langle B^0|\psi(t)\rangle$ and $A_{B\to\bar{B}}(t) = \langle \bar{B}^0|\psi(t)\rangle$. One can therefor interpret $A_{B\to B}(t)$ and $A_{B\to\bar{B}}(t)$ as the probability amplitudes of finding a B^0 and a $\bar{B}^0$, respectively, in the final state after a time t.

Thus in order to find the mixing probability it is necessary to compute the norm of the amplitude in Eq (2.10).

$$\begin{aligned} P_{B^0\to\bar{B}^0}(t) &= |A_{B\to\bar{B}}(t)|^2 \equiv [A_{B\to\bar{B}}(t)]^\dagger \cdot [A_{B\to\bar{B}}(t)] \\ &= \frac{1}{4}[e^{i(M_L+i\frac{\Gamma_L}{2})t} - e^{i(M_H+i\frac{\Gamma_H}{2})t}] \cdot [e^{-i(M_L-i\frac{\Gamma_L}{2})t} - e^{-i(M_H-i\frac{\Gamma_H}{2})t}] \\ &= \frac{1}{4}e^{-(\frac{\Gamma_L+\Gamma_H}{2})t} \cdot \Big[e^{-(\frac{\Gamma_L-\Gamma_H}{2})t} + e^{-(\frac{\Gamma_H-\Gamma_L}{2})t} - \\ &\qquad e^{-i(M_H-M_L)t} - e^{i(M_H-M_L)t}\Big] \end{aligned} \tag{2.12}$$

$$P_{B^0\to\bar{B}^0}(t) = \frac{1}{2}\cdot e^{-\Gamma t}\cdot[1-\cos(\Delta m\cdot t)] \tag{2.13}$$

Where $\Delta m \equiv M_H - M_L$, $\Delta? \equiv ?_H - ?_L$ and $? \equiv (?_H + ?_L)/2$. Due to $\Delta?/? \leq 10^{-2}$ the two $e^{-(\Delta\Gamma/2)t}$ terms in Eq (2.12) was approximated by 1, to get from Eq (2.12) to (2.13)

In a similar fashion one can obtain an expression for the probability of a non-mixed state, i.e. that the final state is a B^0:

$$P_{B^0\to B^0}(t) = \frac{1}{2}\cdot e^{-\Gamma t}\cdot[1+\cos(\Delta m\cdot t)] \tag{2.14}$$

The mixing probability found in Eq (2.13) was in the case of an initial $|B^0\rangle$ state and a final $|\bar{B}^0\rangle$, but it would turn out exactly the same if it was an initial $|\bar{B}^0\rangle$ and a final $|B^0\rangle$ state. So this mixing probability is valid for both B^0 and $\bar{B}^0$ initial state mesons.

In the given situation, with an initial B^0 and two possible final states, B^0 and $\bar{B}^0$, it would be desirable that the time-integrated probabilities add up to unity. But that is not the case, instead $\int\{P_{B^0\to B^0}(t) + P_{B^0\to\bar{B}^0}(t)\}dt = \frac{1}{\Gamma}$. So, in order to normalize the probabilities it is necessary to multiply Eqs (2.13) and (2.14) by a factor $?$.

$$P_{B_q^0\to\bar{B}_q^0}(t) = \frac{1}{2}?_q\cdot e^{-\Gamma_q t}\cdot[1-\cos(\Delta m_q\cdot t)] \tag{2.15}$$

$$P_{B_q^0\to B_q^0}(t) = \frac{1}{2}?_q\cdot e^{-\Gamma_q t}\cdot[1+\cos(\Delta m_q\cdot t)] \tag{2.16}$$

The probability that an initial B_q^0 meson will decay, at a proper time t, as either a $\bar{B}_q^0$ or as a B_q^0, is given by Eqs (2.15) and (2.16), respectively. The interesting thing is that Δm_q gives the oscillation frequency, the larger Δm_q is the larger the frequency is and if Δm_q is zero there would simply be no oscillation. Because of this feature it is common to give results for Δm_q in units of frequency, which is inverse time, instead of energy or mass.

2.3.2 Time-integrated mixing formulas

Because the lifetime of the B meson is very short (section 3.1), compared to the time resolution in macroscopic measuring devices, early analysises of B mixing used a time-integrated method. Defining $x_q \equiv \frac{\Delta m_q}{\Gamma_q}$ and integrating the probabilities of Eqs (2.15) and (2.16), one obtains:

$$\chi_q^m \equiv \int_0^\infty P_{B^0 \to \bar{B}^0}(t)dt = \frac{(\Delta m_q/?_q)^2}{2(1+(\Delta m_q/?_q)^2)} = \frac{x_q^2}{2(1+x_q^2)} \tag{2.17}$$

$$\chi_q^u \equiv \int_0^\infty P_{B^0 \to B^0}(t)dt = \frac{2+(\Delta m_q/?_q)^2}{2(1+(\Delta m_q/?_q)^2)} = \frac{2+x_q^2}{2(1+x_q^2)} \tag{2.18}$$

Now the mixed and un-mixed time-integrated probabilities add up to unity, $\chi_q^m + \chi_q^u = 1$, which they should. The formulas are valid both for B_d mesons (q=d) and for B_s mesons (q=s), but they turn out to be usefull only in the B_d case. The reason for this is that χ_q^m is sensitive only for small mixing.

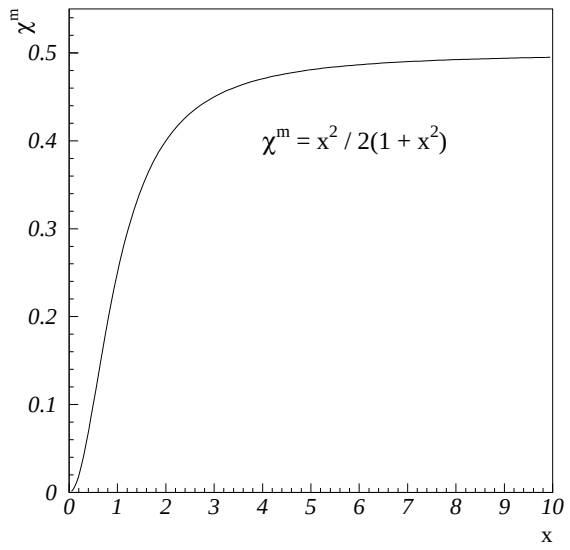


Figure 2.3: The integrated mixing probability χ^m

When $x_q \gg 1$ the system experience *full mixing* and, as figure 2.3 shows, the integrated mixing variable χ_q^m is saturated with a value close to $\frac{1}{2}$. In this situation even large changes in x_q will have minimal effects on the χ_q^m value. On the other hand, when $x_q < 1$, the χ_q^m has a steep gradient and will be very sensitive to even small changes in x_q.

Thus it is clear that the time-integrated method is most sensitive, and thus only useful, for low mixing. In practice $x_q < 1$ is a good criteria for when the time-integrated method works well.

Chapter 3

Experimental methods and measurements

This chapter serves as a little summary of methods used in B physics, to classify B mesons and the sign of the b quarks involved. A brief summary of observed quantities in the B sector, is also given. The chapter ends with a section outlining the physics motivation behind this analysis.

3.1 The B sector

The b quark, also known as bottom or beauty, was announced discovered by Leon Lederman and his team at Fermilab on June 30 1977. They discovered the b quark through the upsilon resonance at 9.46 GeV [13], which is the lowest bound energy state of bottomonium ($b+\bar{b}$ quark).

The b quark [14] has a mass of between 4.1 GeV and 4.5 GeV, which is more than twice the mass of the charm quark and 4 times heavier than a proton. Together with a lighter quark the b quark can form mesons. B mesons have an important common feature: The long lifetime, of the order picoseconds[1] [15].

In section 2.1 the quark structure of neutral B mesons were outlined. The mass of such a meson depends on which quark the b quark is forming the meson with. Because the s quark is slightly heavier than the d quark the B_s meson will be slightly heavier than the B_d meson: $M_{B_s} \approx 5.38$ GeV and $M_{B_d} \approx 5.28$ GeV. This makes it possible to design an experiment with just enough center of mass energy to produce B_d mesons but not B_s. This difference in production threshold was explored in the 1980's, looking for B mixing where it is important to separate the signals from B_d and B_s.

[1] Upon first encounter a picosecond, or 10^{-12} second, may seem like a short time. But it should be compared to the time scale of the strong interaction, in which the B mesons are formed. That scale is of the order 10^{-23} second, hence a B meson typically lives for about 3000 "strong years".

3.2 How to classify a B event

When looking for B mesons in experiments two obvious physical quantities, illustrated in figure 3.1, comes to mind:

1. The large mass M_B
2. The long lifetime τ_B

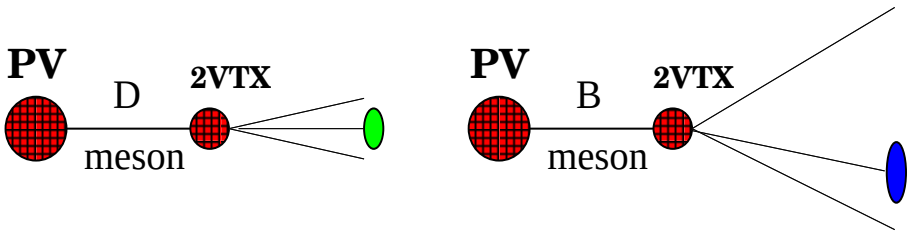


Figure 3.1: Qualitative comparison of collimination of decay tracks from a B meson and a lighter D meson. Both mesons decay at a secondary vertex, but the B meson gives rise to more spread out tracks in its jet than the D.

Due to its long lifetime the B meson will fly a considerable distance [15], $c \cdot \tau_B = 449 \mu m$, from the Primary Vertex (PV). In high energy experiments it can be boosted up to several millimeters, before it decays in a Secondary Vertex (2VTX). This leads to large impact parameters, with respect to the PV, of the particles coming from the B decay.

The large mass, M_B, of the B meson forces the jet from a B decay to spread out like figure 3.1 indicates. The reason is that Nature wants to conserve the *invariant mass*, $P_\mu P^\mu$, where P^μ is the 4-momentum:

$$P_\mu P^\mu = E_{tot}^2 - P_{tot}^2 = M_B^2 \tag{3.1}$$

The B meson is much heavier than the sum of masses of the particles coming from its decay, so in order to conserve the invariant mass the decay-particles are spread out in the jet. Then a larger fraction of each particle momentum will be transverse with respect to the original quark direction, and thus can cancel out large momentum components from other particles in the jet. In this manner the total momentum P_{tot} can be made sufficiently low to conserve the large invariant mass according to Eq (3.1). In the case of a light meson decay the invariant mass is small, hence the total momentum can approach the total energy to a greater extent - which again means that the jet can be more collimated.

Several methods have been developed to classify, or tag as physicists prefer to say, B events. A popular one is based on semileptonic decays, i.e.

a quark decaying to a lepton and someting else. Using energy-momentum conservation it is possible to show that the transverse momentum p_t, with respect to the jet axis, of a lepton from a semileptonic decay, must obey $p_t < \frac{1}{2}m_q$ [16]. Here m_q is the mass of the decaying quark. Thus a larger quark will in general give rise to more isolated leptons, that is leptons with larger transverse momenta. Because a c quark has mass $m_c \approx 1.5$ GeV leptons from a semileptonic c decay should have $p_t < 0.75$ GeV, while from a B decay they should have $p_t < \frac{1}{2}m_b \approx 2.2$ GeV. Thus by requiring for example $p_t > 1.0$ GeV one can inhance a sample with B events.

A common problem with the old tagging methods are that they all lack good efficiencies. Semileptonic tagging only works with semileptonic decays, it is useless in hadronic decays. Thus a more efficient method was sought for. In the recent 2-3 years a highly efficient, but more complex, method called *Lifetime Tag* has been developed.

3.2.1 The Lifetime Tag

The *Lifetime Tag* [17] explores the most obvious feature of a B event, the large track impact parameters, to construct a variable on which one can make continous cuts to obtain exactly the efficiency or sample purity one needs for a given analysis.

The lifetime signed impact parameter d is defined as the shortest distance between the track, when extrapolated back towards PV, and the PV. The sign is positive if the angle between the track and the jet axis is less than $\frac{\pi}{2}$ and negative if not. This is visualized in figure 3.2 where $\alpha > \frac{\pi}{2}$ gives $d < 0$ while in the case of $\alpha < \frac{\pi}{2}$ one gets $d > 0$.

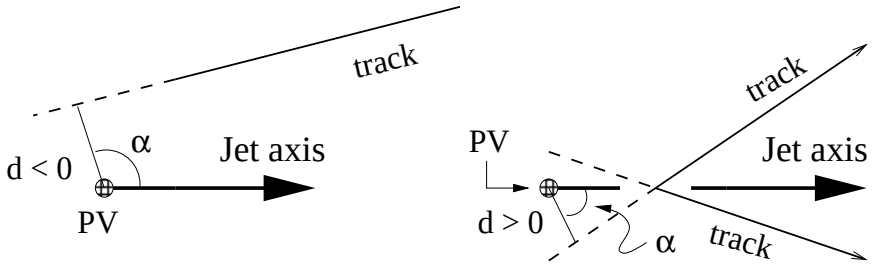


Figure 3.2: Defining the signed impact parameter d.

However, due to finite resolution and flaws in the measuring devices and methods there is an error in the impact parameter. This error will vary with time. In order to find a good variable, independent of how well the measurement was performed, the signed impact parameter is divided by its

error: $S \equiv \frac{d}{\sigma}$, where S is known as the significance and σ is the error in the impact parameter.

With this definition tracks with negative significance are either badly reconstructed (don't belong to the jet) or comes directly from PV. Tracks coming from PV have a 50% chance of being assigned a negative significance, depending on which side of the center coordinates the extrapolated track pass. Hence tracks with $S < 0$ are background in a B analysis.

To construct a method one simply assumes that without B events present the significance distribution, $f(S)$, is symmetric around S=0. $f(S)$ will typically have a peak at S=0 and falling rapidly for large values of S. When B events, with their large positive significance values, are added there will be a surplus of large positive S values as compared to the expected background from the negative S distribution. This surplus will increase with increasing S value, and by integrating it one can form a good variable. Thus when $f(S)$ is properly normalized one can integrate the negative part of the distribution from $-\infty$ to a certain value S_0[2]:

$$P(S_0) = \begin{cases} \int_{S<S_0} f(S)dS & \texttt{if } S_0 < 0 \\ \\ P(-S_0) & \texttt{if } S_0 > 0 \end{cases}$$

The interpretation of $P(S_0)$ is that *it is the probability that a track from PV will have* $S > S_0$. From the definition it is clear that $P(S_0)$ is close to unity for small values of S_0 and close to zero for large.

$P(S)$ is used as the b-tag track probability. Since tracks from a B decay usually have large positive significance values, S_b, the probability, $P(S_b)$, that they originate from PV is small. Hence by cutting hard on $P(S_b)$ one can increase the chances of finding tracks from a B decay.

An even better tagging variable is the N-track probability, P_N, where the track probabilities of all tracks in the event have been combined into:

$$P_N = \prod \cdot \sum_{j=0}^{N-1} \frac{(-\ln \prod)^j}{j!}$$

$\prod \equiv \prod_{i=1}^{N} P(S_i)$ is the product of the N tracks' track probabilities. The P_N variable has the pleasant property that it is uniformly distributed on [0,1]. By a sufficiently low cut on P_N one can obtain purities of more than 90% while a loose cut will give a higher efficiency (but resulting in more contamination from the background). It is by far the best B-tagging method today.

In this thesis the single track b-tag probability $P(S)$ was used because the neural networks were fed with information from one track at a time (see section 6.2.4).

[2]In practice tracks with impact parameters larger than 2mm are cut away to remove kaons and other longlived particles.

3.3 Jetcharge

When a quark decays it will give rise to a debris of particles travelling in approximately the same direction as the quark did before its decay. This debris of particles is called a jet, it can be both very collimated but also very spread out. To reconstruct the sign of the charge of the quark that produced the jet the *Jetcharge method* has been developed.

The idea behind jetcharge is that tracks with high momenta, which are more likely to carry information from the original quark, should count more than softer tracks. Hence in the algorithm the charge of each track is weighted by the momentum of the track. By computing the jetcharge, based on all the tracks from a jet, one should on the average get a value with the same sign as the quark that produced the jet.

In a B mixing analysis all events are $Z^0 \to b\bar{b}$, thus each event can be split in two hemispheres containing one quark jet each. In DELPHI the following algorithm is used to compute the jetcharge of each jet:

$$Q_{jet} = \frac{\sum_j q_j (\vec{p_j} \cdot \vec{e_s})^\kappa}{\sum_j (\vec{p_j} \cdot \vec{e_s})^\kappa} \tag{3.2}$$

Where q_j and $\vec{p_j}$ are the charge and momentum of track j, $\vec{e_s}$ the sphericity axis of the event, which contains the jet, and $\kappa = 0.6$. This structure and choice of κ are based on several years of optimization in DELPHI, and is used in their current B_s mixing analysis [20]. Other experiments use variations of the same formula, ALEPH with an interesting application of rapidity [21].

Given a sample of events one can compute the jetcharge of each of the two hemispheres. By using a cut like $|Q_{jet}| > Q_{cut}$ it is possible to increase the fraction of hemispheres where the sign of the charge is correctly classified. By increasing Q_{cut} the fraction of correctly classified hemispheres will increase, but with the negative effect of reducing the efficiency. As will be made clear in section 5.3 it is important to both have a high fraction of correct classified hemispheres and a high efficiency in a B mixing analysis. This means that one can not cut too hard on the jetcharge. The DELPHI B_s oscillation team currently cuts at $Q_{cut} = 0.10$, which gives them an efficiency of 67.5% and a purity of 68.8% [20].

3.4 Observing B mixing

Because physicists in the early 1980's expected a light top quark (why should the top be so much heavier than the 5 other quarks?) they predicted a small, maybe not observable, mixing in the B_d^0 system. UA1, at CERN, observed a positive signal from $B^0 - \bar{B}^0$ mixing but ascribed most of the effect to the B_s^0 meson, which they could not distinguish from B_d^0.

However, the ARGUS experiment at DESY and CLEO at Brookhaven started running on the ?(4S) resonance of 10.58 GeV. This is below the B_s threshold which means they had just enough center of mass energy to create a B_d meson and its anti-particle, but no B_s mesons. Without the contamination from B_s ARGUS made the first observation of sizable mixing of B_d mesons, which they published in 1987 [18].

Because the mixing in the B_d sector is fairly small, $x_d < 1$, ARGUS used a time-integrated analysis. They used the lepton sign from semileptonic B decays to tag the B flavor in each hemisphere, and then simply count the number of events with two equal lepton signs ($N_{l^+l^+} + N_{l^-l^-} \equiv N_{l^\pm l^\pm}$) and with opposite signs ($N_{l^+l^-}$). A factor $\lambda \approx 1$ was included due to effects from different lifetimes and decay widths of the B mesons. Then ARGUS could calculate [18], [19]

$$\chi_d^m = \frac{N_{l^\pm l^\pm} \cdot (1+\lambda)}{N_{l^+l^-} + N_{l^\pm l^\pm}} = 0.174 \pm 0.053$$

Using Eq (2.17) this leads to $x_d \approx 0.73$, not far from the current value given by the Particle Data Group [11]: $x_d = 0.71 \pm 0.06$.

This unexpectedly large value of mixing in the B_d sector prompted Paolo Franzini to write in his *$B\bar{B}$ mixing: a review of recent progress* [12]: "The simplest interpretation of the ARGUS result in this context is that the top quark is rather heavy". Indeed that turned out to be the case, predicted by B_d^0 mixing almost a decade before Fermilab observed their top events.

After the observation of sizable mixing among B_d mesons experiments started to look for B_s mixing which was expected to be larger. But unfortunately it turns out that x_s is very much larger than x_d, maybe a factor 10 or more. This gives a time-integrated value $\chi_s^m \approx 0.5$, i.e. in a region where it is highly insensitive to variations in x_s. Thus the time-integrated method fails in a B_s analysis.

3.5 B_s mixing - physics motivation

In the B_s sector mixing is much harder to measure, currently there is only a lower limit. For their 1991-94 data DELPHI recently published $x_s > 5.1$, or more precisely $\Delta m_s > 4.6\ ps^{-1}$ at 95% confidence level, in a conference paper [20]. Because of the rapid oscillations it is important to have as much statistics as possible to be able to pin down the oscillation frequency. The time resolution in the detector is also of utter importance in an analysis like this. Figure 3.3 indicates how the B_s oscillations may look like compared to the slow B_d; in plots (A) and (C) the $B_d^0 - \bar{B}_d^0$ curves are given while in (B)

and (D) those for $B_s^0 - \bar{B}_s^0$. The plots were made using

$$P(\zeta) = \frac{1}{2} \cdot e^{-\zeta} \cdot [1 - \cos(x_q \cdot \zeta)]$$

From Eq (2.14) it should be clear that $\zeta = ? \cdot t = \frac{t}{\tau}$, hence the oscillation curves are plotted in units of B lifetime. As one can readily see from plots (B) and (D) the oscillation period for the B_s system is of the order one unit of the B lifetime. To pin down this rapid oscillation the demand for statistics is high.

The full analysis done on the B_s mixing is rather complex and channel dependent. It involves reconstructing the production flavor, using jetcharge, then computing the proper lifetime of the B mesons and finally classifying the decay flavor of the B through its semileptonic decay channels. Refer [19].

The important part to notice is that the B_s analysis depends strongly on the statistics. In appendix A.4 it is shown that the quantity $\sqrt{\epsilon} \cdot (2\rho - 1)$ is proportional to the statistical significance of a signal from oscillations. Thus both the efficicency ϵ and purity ρ of the sample should be as high as possible. In section 3.3 jetcharge was briefly discussed, and the DELPHI efficiency of 67.5% for the jetcharge method listed.

Since a B_s analysis depends on this efficiency one may pose the question "Is it possible to improve the efficiency of classifying the production charge of b quarks without jeopardizing the purity?". Since the jetcharge itself is so highly optimized it is unlikely that one can improve the figures much by adjusting the algorithm in equation (3.2). But maybe a neural network can perform better? Given that neural networks can access more information than the momentum and charge, used in the jetcharge, this question was interesting enough to trigger this analysis.

Thus the main goal of this thesis was to check if it is possible to improve the jetcharge values, the purity and efficiency of classifying the production flavor of b quarks, by using an Artificial Neural Network.

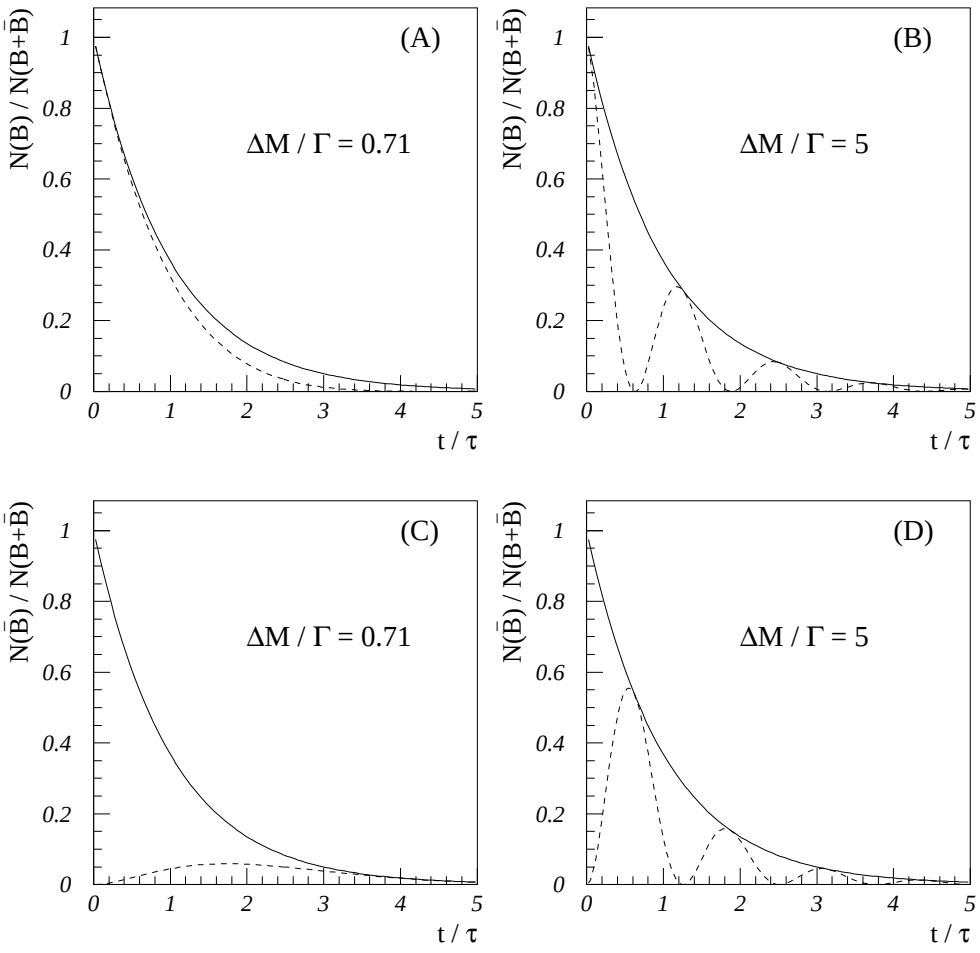


Figure 3.3: The dotted lines are the oscillation curves and the continous lines the damping exponential function. In these plots the initial sample contained only B^0 mesons. Plots (A) and (B) show the development of the fraction of B^0 in the sample, while plots (C) and (D) show the development of the fraction of $\bar{B}^0$. The dimension along the abscissa is given in units of B meson lifetime, while $\frac{\Delta M}{\Gamma}$=0.71 is the situation for $B_d^0 - \bar{B}_d^0$ mixing and $\frac{\Delta M}{\Gamma}$=5 is a likely lower limit for the situation in $B_S^0 - \bar{B}_S^0$ mixing.

Chapter 4

Introduction to neural networks

A brief introduction to the concept of neural networks, with emphasis on feed-forward structures that use supervised learning, is given. The specific choices made for this thesis are listed in appendix B.

4.1 Historical motivation

The idea of using networks to process information came about in the 1940's, though not strictly developed until medical scientists coined their model of the human brain by the name *neural network*, in the 1950's. This model of the brain turned out to be too simple, but the idea of neural networks was born. In the 1960's the interest in neural networks fell dramatically when scientists came to the conclusion that simulations of neural networks would not be possible with the computer resources of that time. But with the birth of semiconductor technology and microchips the computing power development went into overdrive, bringing back the old ideas of simulating neural networks.

The familiar personal computers or workstations are all part of a class of computers commonly called Von Neumann machines. The traditional Von Neumann machine has two distinct properties:

- Sequential execution of instructions from a store containing instructions and data.
- Most of the store is empty most of the time.

Up to know these properties have been less critical than other factors. But in the future, with the need for realtime simulations and large amounts of data processing, these properties will turn into a bottleneck. At the future LHC experiments one will have to do fast trigger decisions and crude online measurements of millions of bytes of information, in a matter of microseconds. Instead of having a Von Neumann machine, processing the data bit by bit, a solution with massive parallel processing is needed.

In environments, like the LHC, the neural network technology offers a possible solution. It can store the information almost uniformly among the weights in the network and due to its structure the neural network acts like a huge cluster of parallel processing Von Neumann machines.

4.2 Neural network architecture

Neural networks consist of many small computing machines called neurons or nodes. The internal connections between the neurons can have some resistivity, often called weights. If the connection between two neurons are weighted by a factor zero the connection is said to be broken. Each neuron contains an activation function which, based on the total input to the neuron, computes the neuron output.

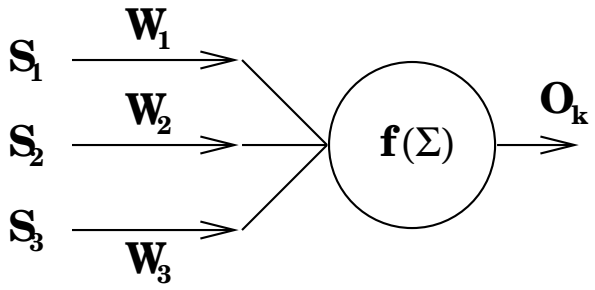


Figure 4.1: O_k is the output from neuron k, based on the three input signals (S_1, S_2 and S_3) with corresponding weights (W_1, W_2 and W_3).

Figure 4.1 illustrates how the neuron works. First the weighted input signals, to the neuron, are summed up: $\sum = \sum_{j=1}^{n}(S_j \cdot W_j)$. This sum is passed along to the activation function f. This function will form the output from the node, in some cases it incorporates a threshold so that if $\sum$ is too small the neuron will give zero output.

4.3 Feed-forward networks

There are many classes of neural networks and the simplest of them all is the feed-forward structure used in this thesis. The reason why this structure is so popular is partly because it is very easy to implement in computer software and partly because it allows supervised learning. In supervised learning the net is required to give some desired output for a certain class of input patterns. This is very useful in problems where good simulations exist.

The feed-forward structure is simple in the sense that it is layered. When a signal pass through the net it always pass from one layer to the next, not

to other neurons in the same or the previous layer. This is illustrated in figure 4.2. In a feed-forward structure the first layer of neurons is called the input layer, that is the layer where the input patterns based on the data sample are fed in. Then follows one ore more hidden layers, as the name indicates these layers can not be accessed from outside the net. The final layer is the output or outer layer, which may contain one or more output neurons. This is where the network decision, for a given input pattern, can be read out.

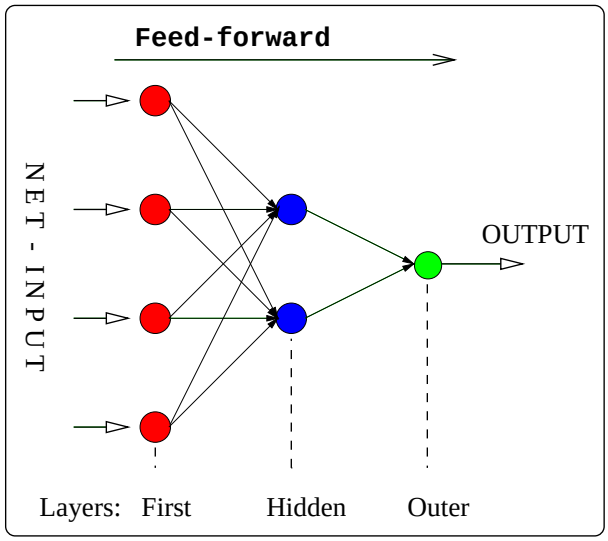


Figure 4.2: The structure of a feed-forward network

Because the input to the network is known, and the activation functions are chosen by the user, the total output can be calculated as a function of the weights. Thus by changing the internal weights of the network one can adjust the total output to any desired value. This is the basic concept behind the supervised learning.

4.4 The error measure

When training a neural network the main concern is with the average squared training error, χ^2:

$$\chi^2 = \frac{1}{2 \cdot N_p} \cdot \sum_{p=1}^{N_p} \sum_i \left(t_p^i - o_p^i\right)^2 \tag{4.1}$$

where N_p is the number of training patterns, t_p^i is the desired target value, for pattern p and output neuron i, and o_p^i the true output value from output neuron i for the same input pattern p.

4.5 Training a feed-forward network

When the activation functions are known the output from the network, for a given input pattern, depends only on the weights. The problem of training the network is then reduced to finding a set of weights that minimize Eq (4.1).

But one is looking for a network solution that can recognize patterns it has not seen before, it should be able to generalize data. Thus one can not use χ^2=0 as a training criteria, this would be an overfit: The net would only be able to recognize patterns from the training sample. Instead one must look for a solution of the weights that gives a global minimum of χ^2, i.e. that approximates $\frac{\partial\chi^2}{\partial\vec{w}}$=0, where $\vec{w}$ is the vector of weights. Such a solution will recognize the general features of the test sample.

4.5.1 Back-propagation

To minimize χ^2 one use a gradient descent method which is incorporated in the back-propagation algorithm. The gradient $\vec{\nabla}\chi^2 = \frac{\partial\chi^2}{\partial\vec{w}}$ is used to update the weights in order to reach a minimum of χ^2:

$$\Delta\vec{w}_{t+1} = -\eta\vec{\nabla}\chi^2 + \alpha\Delta\vec{w}_t \tag{4.2}$$

Where the t subscript indicates the time order. η is known as the learning strength, it gives the step length in the weight update, while α is a momentum term which insures stable learning by bringing in the previous weight update.

The back-propagation algorithm can be listed in a cockbook manner:

1. Calculate χ^2 when feeding the signal from the input pattern forward through the network
2. Calculate the gradient in a backward sweep through the network
3. Modify the weights according to: $\vec{w}_{t+1}$=$\vec{w}_t$+$\Delta\vec{w}_{t+1}$, where $\Delta\vec{w}_{t+1}$ is given by Eq (4.2)
4. Calculate χ^2 as in point 1 but with the new weights. If it gets worse lower η until χ^2 improves
5. Go to 2

4.5.2 The input patterns

The input patterns must be chosen so that they display the particular features one would like the net to learn. They are prepared in an N-dimensional array which are fed into the N input neurons of the input layer. It is important

to limit the number of variables, used in the input patterns, since the actual training of the neural network is very time consuming. Finally one should make sure that the input variables are normalized, to avoid saturating the activation functions.

4.5.3 When to stop training

There is no way to know a priori how much a network should be trained. If it is trained too short it will not have had time to learn the general features of the training sample, but if it is trained too long it will start to specialize on the training sample. So one must try to find a golden mean, thus one should monitor the training error development to see if it reaches a minimum value. After a training session it is important to have a testing sample on which the net can be tested. This will reveal if the net can generalize what it learned from the training sample.

Chapter 5

Preparing for the analysis

This chapter only briefly describes all the work that was put into extracting and checking the simulated data, trying out different quality cuts and setting up the necessary apparatus in order to compare the different outputs.

5.1 Extracting and checking the data

Simulated $Z^0 \rightarrow b\bar{b}$ events, stored in DST format at Shift (the DELPHI offline area), were used as data for this analysis. The data were generated by a Monte Carlo algorithm with full DELPHI detector response simulated, known as DELSIM [26]. A Skelana[27] routine was used to extract the data which then were stored in column-wise Ntuples to make cut manipulations and tests easier. During the extraction three quality cuts were imposed to remove badly reconstructed tracks and tracks with signatures compatible with γ conversions or V^0 candidates:

1. A track was only accepted if the error, σ_E, on the reconstructed energy, E, was less than 100% i.e. that $\sigma_E < E$

2. Loose cuts on γ conversion tags, from ELECID, imposed to remove γ conversions (gave 85% efficiency and 1.6% misidentifications)

3. A tight V^0 tag was used to identify and remove V^0 candidates:

   (a) The angle, in the xy plane, between the V^0 momentum and the line joining PV and 2VTX was required to be less than $(0.01+\frac{0.02}{p_t})$ rad. Here p_t, measured in GeV, is the transverse momentum with respect to the beam axis

   (b) The radial separation, in the xy plane, of PV and 2VTX was required to be greater than 1σ

   (c) The χ^2 probability of the fit to a 2VTX had to be larger than 0.01

Each event was split into two hemispheres by a plane normal to the sphericity axis of the event (figure 5.1) so that the b quark ended up in one hemisphere and the $\bar{b}$ in the other. Per definition all tracks with a momentum component parallel to the sphericity axis were contained in hemisphere 1 and those that were anti-parallel in hemisphere 2. The sphericity [28] is essentially a measure of the summed P_t^2 with respect to the event axis. A two jet event has $S \approx 0$ while in isotropic events $S \approx 1$.

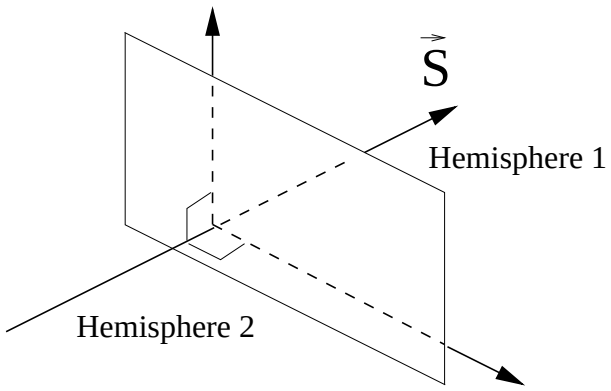


Figure 5.1: A plane normal to the sphericity axis $\vec{S}$ splits the event into two hemispheres, with one B meson in each.

From the simulated data a total of 57261 events were accepted, 34700 of them were stored as a training sample and the remaining 22561 as a test sample. With the imposed quality cuts, especially the σ_E cut, most of the wrongly reconstructed tracks, with momentum above the beam energy, were removed. But around 300 tracks with too high momentum survived the quality cuts, they had to be treated with special care during the training and testing of the nets (see section 6.2.2).

5.2 Defining the hadronic selection

A neural network trained to recognize B and $\bar{B}$ patterns should be able to sort out the common hadronic features of $Z^0 \rightarrow b\bar{b}$ events, making hadronic cuts on the data sample superfluous. To test the neural networks for this ability it was necessary to avoid using hadronic selection criterias during the extraction of the simulated data. Instead the training and testing routines were implemented so that the hadronic cuts could be switched on and off as desired. With that solution it was possible to test how hadronic cuts affected the neural network performance, when the cuts were imposed on the data samples and when not.

The following hadronic criterias were used in this analysis:

1. At least 5 charged tracks in each hemisphere
2. Each accepted track must have a momentum greater than 200 MeV
3. The momenta of wrongly reconstructed tracks are set to 10 GeV

Studies of the momentum distribution of Z^0 decays shows that the average high-momentum track in multihadronic events has a momentum of around 10 GeV. Thus if a momentum value turns out to be above the beam energy, due to bad track reconstruction, a good value to use instead is 10 GeV. This justifies the third hadronic criterion.

Table 5.1 shows the percentage of tracks and hemispheres surviving each of the two first hadronic criterias. The third criterion will only rescale the momentum, in case it is too high, and thus will not affect the efficiency.

Criterion	**Tracks**	**Hemispheres**
1: ntrack.ge.5	99.0%	96.3%
2: p(trk).gt.0.2	86.3%	99.9%
1 + 2	85.4%	96.3%

Table 5.1: Data surviving the hadronic selection

34700 events times two hemispheres result in 69400 training patterns, but with 96.3% surviving the hadronic criterias only 66000 were used during training. In the few cases where the hadronic criterias were not imposed on the sample 69000 training patterns were used.

The 22561 events in the test sample amounted to 45122 test patterns, but in case the hadronic criterias were imposed only 43534 patterns survived.

5.3 How to compare the results

This section outlines a few quantities useful when comparing the results from the various nets and jetcharge, it also gives a brief description of how the output is treated.

5.3.1 Defining three quantities ϵ, ρ and A_{max}

To be able to compare the performance of different neural networks, with each other and with the standard Jetcharge method, three quantities were defined:

1. $\boldsymbol{\epsilon}$: The mean tagging efficiency is the fraction of the B and $\bar{B}$ patterns that have been classified (right or wrong) above a given cut value on the output. Since no distinction between B and $\bar{B}$ is made this is a mean value. ϵ will be refered to as *efficiency*.

2. $\boldsymbol{\rho}$: The mean tagging purity, a.k.a. mean correct tagging efficiency, is the fraction of the classified events that are correct classified. Again no distinction made between B and $\bar{B}$ so this is also a mean value. Hereafter ρ will be refered to as *purity* for simplicity.

3. $\boldsymbol{A_{max}}$: DELPHI refer to $A \equiv \sqrt{\epsilon} \cdot (2\rho - 1)$ as "proportional to the statistical significance of a signal from oscillations", where ρ and ϵ are those defined in points 1. and 2. (see appendix A.4). Because ρ and ϵ are functions of the cut, Q_{cut}, on the output distributions so will A. The maximum of this function, $\boldsymbol{A_{max}}$, is an important quantity in B mixing and hence used for comparison in this analysis.

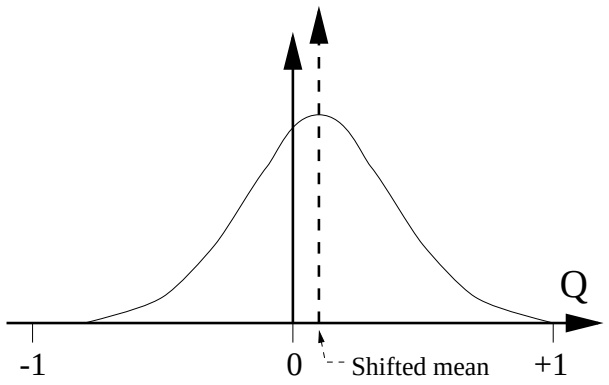


Figure 5.2: DELPHI jetcharge distribution (exaggerated shifted mean)

As figure 5.2 indicates the DELPHI jetcharge distribution has a shifted mean, $|Q_{mean}^{shifted} - 0.015| = 0$. This is due to differences in the detector response for negatively and positively charged particles. A zero bin, labelled *nzero*, corresponding to this shift is calculated.

Because all the tests are done on simulated data the correct b flavor is known ahead, then it is possible to plot the true distribution from the net for a given flavor. A routine named `abscut.f` was written. It takes the $B - \bar{B}$ information as input and returns the ϵ and ρ information, in the form of two histograms.

`abscut.f` is implemented to be very robust: It checks that the two input histograms exists and that they have an equal number of bins (which is required to make sense when calculating ϵ and ρ). The input histograms are

allowed to have any number of bins up to 1000, `abscut.f` will compute how many bins the output histograms should have based on the input histograms. From the ϵ and ρ information it also computes A_{max} and gives the value of ρ for ϵ=1. See program listing, appendix C.3.

5.3.2 Preparing the B and $\bar{B}$ information

B mesons contain a $\bar{b}$ quark, with charge $+\frac{e}{3}$, thus jetcharge from a B should be positive on the average. For $\bar{B}$ mesons the jetcharge distribution should have a negative mean value due to the $-\frac{e}{3}$ charge of the b quark. The netcharge is required to follow the same rules, so when a B pattern is fed into the network the output should be positive and for $\bar{B}$ negative.

Similar to the jetcharge distributions it is possible to construct netcharge distributions from a neural network output. Because the B-flavor of the pattern that caused a specific output is known it is possible to plot the B and $\bar{B}$ distributions of the netcharge.

The first thing `abscut.f` does is to integrate the B and $\bar{B}$ distributions, from -1 to +1. The integrated B information is stored in an array named `bsum` and the integrated $\bar{B}$ in `bbsum`. For compactness reasons these two arrays will be denoted Σ_b and $\Sigma_{\bar{b}\bar{b}}$ respectively. Then for a given entry in $\Sigma_{\bar{b}\bar{b}}$ one will find the integrated value of the $\bar{B}$ distribution from -1 up to a limit on the netcharge corresponding to that entry. Similar for Σ_b.

To estimate the error in each entry of the Σ_b and $\Sigma_{\bar{b}\bar{b}}$ arrays Binomial statistics was used. This is shown in more detail in appendix A.1. Based on Eq (A.3) the errors should be:

$$\left.\begin{aligned} \sigma_b(\delta) &= \sqrt{\frac{n_b(\delta)\cdot[N_b - n_b(\delta)]}{N_b}} \\ \sigma_{\bar{b}\bar{b}}(\gamma) &= \sqrt{\frac{n_{\bar{b}\bar{b}}(\gamma)\cdot[N_{\bar{b}\bar{b}} - n_{\bar{b}\bar{b}}(\gamma)]}{N_{\bar{b}\bar{b}}}} \end{aligned}\right\} \tag{5.1}$$

Where N_b is the total and $n_b(\delta)$ the integrated number of B patterns up to bin number δ in the B distribution. Similar for $\sigma_{\bar{b}\bar{b}}$. Thus $\sigma_b(\delta)$ is the error in $\Sigma_b(\delta)$ and $\sigma_{\bar{b}\bar{b}}(\gamma)$ the error in $\Sigma_{\bar{b}\bar{b}}(\gamma)$.

5.3.3 Right and wrong tagging

To compute the tagging purity and efficiency it is necessary to know how many B and $\bar{B}$ patterns that were classified right and wrong. From figure 5.3 the number of right tags can be found by counting the number of B entries above $+Q_{cut}$ and the number of $\bar{B}$ below $-Q_{cut}$, since the output should be positive for a B input and negative for $\bar{B}$. The number of wrong tags are then the number of B below $-Q_{cut}$ and $\bar{B}$ above $+Q_{cut}$.

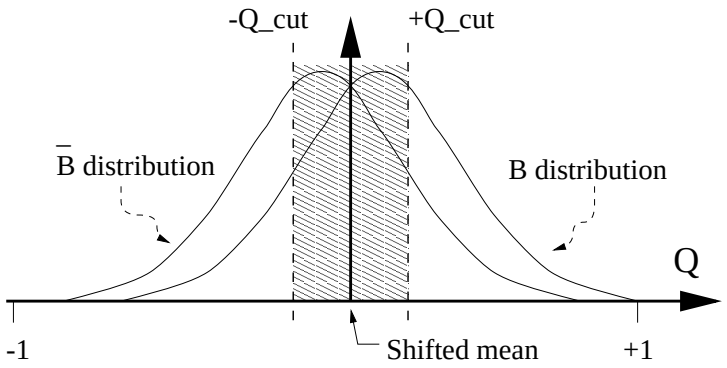


Figure 5.3: Illustration of the B and $\bar{B}$ distributions. The shifted mean is identified by the *nzero* bin of the two distributions.

To find the number of right and wrong tags `abscut.f` calculates how many bins away from *nzero* the cuts are, for example k bins. Then $-Q_{cut}$ is identified as bin number $\beta_{lo} = nzero - k$ and $+Q_{cut}$ as $\beta_{hi} = nzero + k$. The information corresponding to the cut values are then given by the two integrated arrays using β_{lo} and β_{hi} as indexes.

Because the integration is from -1 to +1 the number of patterns below $-Q_{cut}$ can be found directly from the Σ_b and Σ_{bb} arrays with β_{lo} as index. The number of patterns classified above the $+Q_{cut}$ is the difference between the total number of patterns and the number in the cut bin. The total number is simply read from the last position, Ω, in both arrays.

- Number of wrong classified B patterns: $\Sigma_b(\beta_{lo})$
- Number of right classified $\bar{B}$ patterns: $\Sigma_{bb}(\beta_{lo})$
- Number of wrong classified $\bar{B}$ patterns: $\Sigma_{bb}(\Omega) - \Sigma_{bb}(\beta_{hi} - 1)$
- Number of right classified B patterns: $\Sigma_b(\Omega) - \Sigma_b(\beta_{hi} - 1)$

The digit, 1, must be included to count the cut bin on the positive side $(+Q_{cut})$ because the algorithm integrates from the negative side.

5.3.4 The mean tagging efficiency ϵ

The mean tagging efficiency $\epsilon(q)$, where $q = Q_{cut}$, is defined as *the ratio of the number of patterns that are classified, for a given cut, and the total number of patterns.* In other words how large fraction of the patterns that has been classified for a given cut on the netcharge (or jetcharge). By using

the four classification expressions from section 5.3.3 one can write down an expression for the efficiency:

$$\epsilon(q) = \frac{\Sigma_b(\beta_{lo}) + \Sigma_{bb}(\beta_{lo}) + [\Sigma_b(\Omega) - \Sigma_b(\beta_{hi} - 1)] + [\Sigma_{bb}(\Omega) - \Sigma_{bb}(\beta_{hi} - 1)]}{\Sigma_b(\Omega) + \Sigma_{bb}(\Omega)}$$

$$\Downarrow$$

$$\epsilon(Q_{cut}) = \frac{\Sigma_b(\beta_{lo}) + \Sigma_{bb}(\beta_{lo}) + N - \Sigma_b(\beta_{hi} - 1) - \Sigma_{bb}(\beta_{hi} - 1)}{N} \tag{5.2}$$

Where $N = \Sigma_b(\Omega) + \Sigma_{bb}(\Omega)$ is the total number of patterns.

$\epsilon(q)$ is a function of Σ_b and Σ_{bb}, which errors are known from Eq (5.1). By computing the total differential one can find an expression for the error in ϵ. In appendix A.2 this is shown in detail, using Eq (A.10) the error, σ_ϵ, in the efficiency is

$$\sigma_\epsilon(Q_{cut}) = \frac{1}{N}\sqrt{\sigma_b(\beta_{lo})^2 + \sigma_{bb}(\beta_{lo})^2 + \sigma_b(\beta_{hi} - 1)^2 + \sigma_{bb}(\beta_{hi} - 1)^2} \tag{5.3}$$

5.3.5 The mean tagging purity ρ

The mean tagging purity, $\rho(q)$, is defined as *the ratio between the number of correct classified patterns and all classified patterns, for a given cut* $q = Q_{cut}$. Another common name for this quantity is correct tag efficiency.

From section 5.3.3 the number of wrong classified patterns:

$$\Sigma_{wr}(q) = \Sigma_b(\beta_{lo}) + [\Sigma_{bb}(\Omega) - \Sigma_{bb}(\beta_{hi} - 1)] \tag{5.4}$$

The number of correct classified patterns:

$$\Sigma_{ri}(q) = \Sigma_{bb}(\beta_{lo}) + [\Sigma_b(\Omega) - \Sigma_b(\beta_{hi} - 1)] \tag{5.5}$$

The number of classified patterns is the sum of $\Sigma_{ri}(q)$ and $\Sigma_{wr}(q)$, thus:

$$\rho(q) = \frac{\Sigma_{ri}(q)}{\Sigma_{ri}(q) + \Sigma_{wr}(q)} \tag{5.6}$$

$$\Updownarrow$$

$$\rho(Q_{cut}) = \frac{\Sigma_{bb}(\beta_{lo}) + [\Sigma_b(\Omega) - \Sigma_b(\beta_{hi} - 1)]}{\Sigma_b(\beta_{lo}) + \Sigma_{bb}(\beta_{lo}) + N - \Sigma_{bb}(\beta_{hi} - 1) - \Sigma_b(\beta_{hi} - 1)} \tag{5.7}$$

In the last equation N is used instead of $\Sigma_b(\Omega) + \Sigma_{bb}(\Omega)$, which is the total number of patterns (both tagged and untagged). To estimate the error, $\sigma_\rho(Q_{cut})$, in $\rho(Q_{cut})$ one progress just like for the ϵ (section 5.3.4). Refer to appendix A.3 for details, Eq (A.15) gives:

$$\sigma_\rho(Q_{cut}) = \frac{\sqrt{{\Sigma_{wr}}^2 \cdot [\sigma_b(\beta_2)^2 + \sigma_{bb}(\beta_1)^2] + {\Sigma_{ri}}^2 \cdot [\sigma_b(\beta_1)^2 + \sigma_{bb}(\beta_2)^2]}}{N^2} \tag{5.8}$$

The following abbreviations are used: $\beta_1 = \beta_{lo}$ and $\beta_2 = \beta_{hi} - 1$.

5.4 Final comments before the analysis

In experimental physics *one* σ usually means *one standard deviation away from the statistical mean value.* With a Gaussian distribution the standard deviation is defined as the square root of the variance, thus one would expect 68.26% of the events to be within one σ from the mean value, 95.44% to be within two σ of the mean value etc.

In this thesis Binomial statistics were used, the error for a given quantity was taken to be the square root of the expected variance. If one use the σ-notation from Gaussian statistics the errors listed in this thesis amounts to one σ. Thus if a measurement is $\mathcal{N}\sigma$ lower than another measurement the difference is $\mathcal{N}$ times as large as the error listed for the measurement.

This chapter has given a brief description of how the simulated data were extracted and prepared, a few important quantities have also been defined.

- From Eq (5.1) the error for a given bin of Σ_b and Σ_{bb}, the integrated B and $\bar{B}$ distributions, can be computed
- Eqs (5.2) and (5.3) give the mean tagging efficiency and its error
- Eqs (5.7) and (5.8) give the mean tagging purity and its error

All these equations are implemented in `abscut.f`, a semi-robust Fortran 77 subroutine, that should be called from PAW. It takes the $B - \bar{B}$ distributions as input and returns purity and efficiency information.

Chapter 6

The analysis, part 1

The main approach of the analysis is displayed in this chapter, including input variable definitions and test results from the various net configurations.

6.1 Where to start?

By some strange analogy with the training of a neural network the analysis started out by simple trial and error, there was a lot of unknown territory to cover. So instead of following a straight line to a goal the analysis was subject to many changes, during the course of time, to sample as much of the parameter space as possible in search for the optimal solution.

Assuming that Jetnet is a robust package most of its default values were used throughout this analysis, the main emphasis was put on the learning parameters (section B.1) and the normalization of the input variables.

6.2 Defining the input variables

Early on it was clear that a cut had to be made on the number of tracks, from each hemisphere, used as input to the net. The reason is that the number of input neurons equals the number of tracks times the number of input variables associated with each track.

Most of the information from the B production is carried by a few hard tracks, i.e. those with highest momentum. By sorting all the tracks in a hemisphere according to their momenta one can define n_{hard} *as a certain number of hard tracks* and sum up the remaining in one *pseudotrack*. Then it is possible to use a constant number of inputs to the net, no matter how many tracks a hemisphere contains. During the analysis n_{hard} could be set to different values, thus varying the number of input tracks to the nets.

Let n_{inp} be the number of extracted input variables from each track, then the total number of inputs, from each hemisphere, to the net is given by $N_{tot}=(n_{hard}+1)\times n_{inp}$, where "+1" comes from the combined pseudotrack.

The number of input neurons in the net must equal N_{tot} so it is important to limit the number of input variables, n_{inp}, and hard tracks, n_{hard}, in order to reduce the size of the net and thus training time and RAM-usage.

6.2.1 Charge q

Together with the momentum the charge of a track is used, in the standard jetcharge method, to reconstruct the production charge of the jet with which the track is associated. Hence the charge and momentum of a track were obvious input variables to the neural nets.

A great advantage with charge is that it is either -1 or +1, neutrals are not used, so for the hard tracks it was used directly as input. For the pseudotrack a jetcharge calculation was done, using the standard formula:

$$Q_{hem} = \frac{\sum_j q_j (\vec{p_j} \cdot \vec{e_s})^\kappa}{\sum_j (\vec{p_j} \cdot \vec{e_s})^\kappa} \tag{6.1}$$

In which the index j runs over all the non-hard tracks, q_j is the charge and $\vec{p_j}$ the momentum of track j. $\vec{e_s}$ is the sphericity axis of the event which the hemisphere is part of. DELPHI use $\kappa = 0.6$ in their latest B_s^0 paper [20].

6.2.2 Momentum p

Unlike charge the momentum can vary over a large range, from almost zero to multiple GeV. In fact, due to the finite detector resolution, resulting in a poorly reconstructed track, the momentum can be of the order 100 GeV. This is of course an unphysical momentum value and most of the tracks with such values were removed by the quality cuts, listed in section 5.1, during extraction from the DST's. But a few survived, so during training and testing all remaining tracks, with momentum above the beam energy, were scaled to 10 GeV.

For a standard hard track, as well as those with a rescaled momentum value, the momentum was normalized to the LEP1 beam energy, 45.6 GeV, before used as input. For the pseudotrack a combined momentum variable, inspired by the momentum term in the jetcharge, was defined as

$$P_{soft} = \sum_j (\vec{p_j} \cdot \vec{e_s})^\kappa \tag{6.2}$$

Where the sum runs over all the non-hard tracks. This variable was also normalized to the beam energy before used as input to the neural nets.

6.2.3 Transverse momentum p_t

The transverse momentum of a track in a B event is an important quantity, which has been explored for years (see section 3.2). It gives rise to the

large impact parameters which form the basis for the B-tagging mentioned in section 6.2.4. To extend the B-patterns, used as input, the p_t information was added to see if that made it easier for the neural nets to form a good decision surface.

The p_t is defined as the transverse component of the momentum with respect to the sphericity axis of the event, and it is normalized to the beam energy. For the pseudotrack this variable was defined:

$$P_{t,soft} = \sum_j (\vec{p}_{t,j})^\kappa \tag{6.3}$$

Like for P_{soft} the sum is over all non-hard tracks and normalized to the beam energy.

6.2.4 B-tag probability

The final variable used in this analysis explores the track probability, P_{track}, from the B-tagging described in section 3.2. It should be small for tracks in a B event. Like any other probability this variable is normalized, but a logarithmic transformation was still used to explore the region close to 0:

$$\log_b = \frac{\log(P_{track})}{\log(P_{min})} \tag{6.4}$$

P_{min}=10^{-10} was the minimum allowed value for the track probability, if P_{track} was smaller it was set to P_{min}. The reason for this was partly to make $\log_b$ normalized, and partly because $P_{track} < 10^{-10}$ is essentially zero (it is nonsense to operate with even lower track probabilities). With this definition $\log_b \in [0,1]$, close to 1 for tracks that are likely to come from a B.

For the pseudotrack a variable, inspired by the definition of the Ntrack probability in section 3.2, was defined: $\prod = \prod_j(P_j)$ where P_j is the B-tag probability of track j. Instead of using this product directly the same logarithmic transformation, as for the hard tracks, was used and the result was averaged over the number of soft tracks:

$$\log_{soft} = \frac{\log(\prod_j(P_j))}{n_{soft} \cdot \log(P_{min})} = \frac{\sum_j \log(P_j)}{n_{soft} \cdot \log(P_{min})} \tag{6.5}$$

Where n_{soft} is the number of soft, or non-hard, tracks and P_{min} has the same value as for the hard tracks.

6.3 Training an Artificial Neural Network

In section 6.2 four potential input variables were outlined. In section 5.3 a description of how to treat the output from the neural nets, for comparison

measurements, was given. The logical next step would be to make an estimate of how many hidden layers, hidden neurons and output neurons the problem of separating B and $\bar{B}$ requires. Unfortunately it is not possible to calculate those numbers a priori, leaving trial and error as the only alternative.

However the JetNet 3.0 write-up, and several papers on the topic, clearly states that for most of the problems in HEP one hidden layer should be sufficient to encode a solution. This was used to narrow down the number of possible combinations of net structures: All nets used in this analysis had a structure with one hidden layer.

In a DELPHI analysis using an ANN to separate quark flavors [24] they had several distinct classes, one for each quark flavor. For each class they assigned an output neuron which should "fire" only if a pattern from that class was recognized. In the current analysis the aim was to classify the sign of the charge of the b quark in the B meson. One can not really speak of widely separate classes, thus only *one* output neuron was used. This output neuron was required to give +1 for a B and -1 for a $\bar{B}$ pattern, corresponding to a $\bar{b}$ and a b quark. Then the sign of the output from the net should match the sign of the b quark inside the meson and thus justify the name netcharge for the net-output, using the analogy with jetcharge.

It is important, at this point, to stress that the training and testing of the neural networks were done with two different samples, one training and one testing sample. It would be pointless to test a net with data it recognizes from the training. This was implemented in the training and testing routines at an early stage, so all test results in this analysis are really based on the generalization performance of the different nets.

Throughout this analysis *one training cycle* will mean *one loop through the training sample.* In `JetNet` the weights are updated every 10th pattern and using the 66000 patterns in the training sample, that survived the cuts in section 5.2, it turns out that 1000 training cycles equals 6.6 million updates of the weights!

6.4 Using the p and q information

To be able to test how well the nets behave, that the training works and the program is free of serious bugs, it is important to have means of comparing the results. By starting very simple, using only p and q information as input, the idea was to compare the output with that coming from jetcharge by using the formalism outlined in section 5.3: For a given cut on the netcharge the efficiency and purity of classifying the input-patterns, based on the testing sample, can be calculated and compared with what the jetcharge gives for the same cuts when it has been computed for the same data sample.

Figure 6.1 shows the total jetcharge distribution and the corresponding B and $\bar{B}$ distributions from the jetcharge. The hadronic cuts of section 5.1

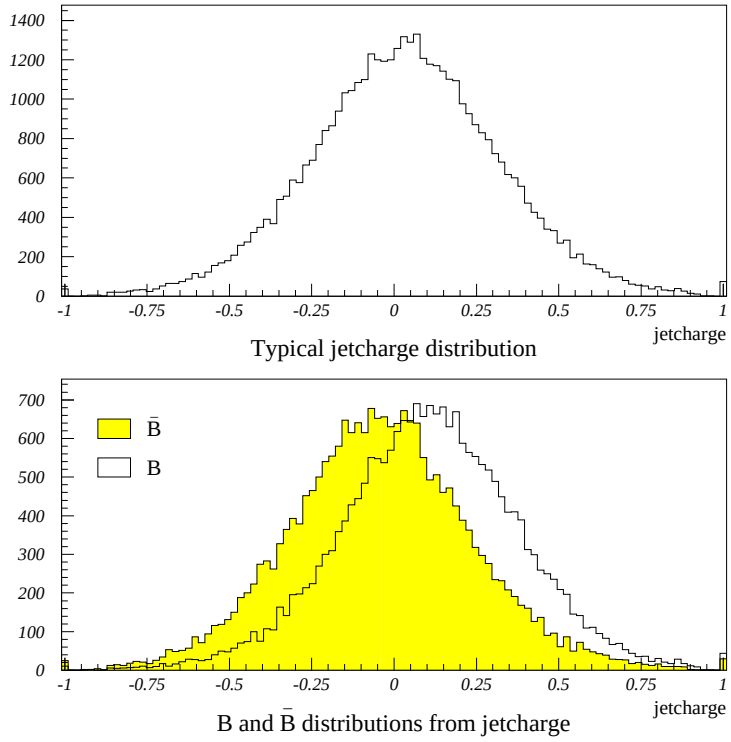


Figure 6.1: Distributions based on Jetcharge

were imposed on the test sample before the DELPHI definition of jetcharge, Eq (6.1), was applied to the remaining tracks in each hemisphere.

After extracting the B and $\bar{B}$ distributions the **abscut.f** routine was used to determine the mean tagging purity and efficiency as functions of the cuts on these distributions. The results obtained were $\rho(\epsilon{=}1) = 0.619$, that A_{max} occurs for $Q_{cut}{=}0.13$, with an error of 0.01 due to finite bin width, and that $\rho(Q_{cut}{=}0.13) = 0.660$ and $\epsilon(Q_{cut}{=}0.13) = 0.644$. This is in fair agreement with the DELPHI paper [20] which list $\rho(\epsilon{=}1) = 0.635$, that A_{max} occured for $Q_{cut}{=}0.10$ and that $\rho(Q_{cut}{=}0.10) = 0.688$ and $\epsilon(Q_{cut}{=}0.10) = 0.675$. The small differences are probably due to DELPHI using a few more hadronic selections than those used here, like cuts on the reconstructed energy.

As the test sample was found to give good results the next step was to train a few nets using p and q information. The choice fell on a structure with 10 input-tracks, amounting to 9 hard and one pseudotrack. This required a total of 20 input neurons, 10 neurons were assigned to the hidden layer. This 20-10-1 net was trained for 1000 cycles and 15000 cycles.

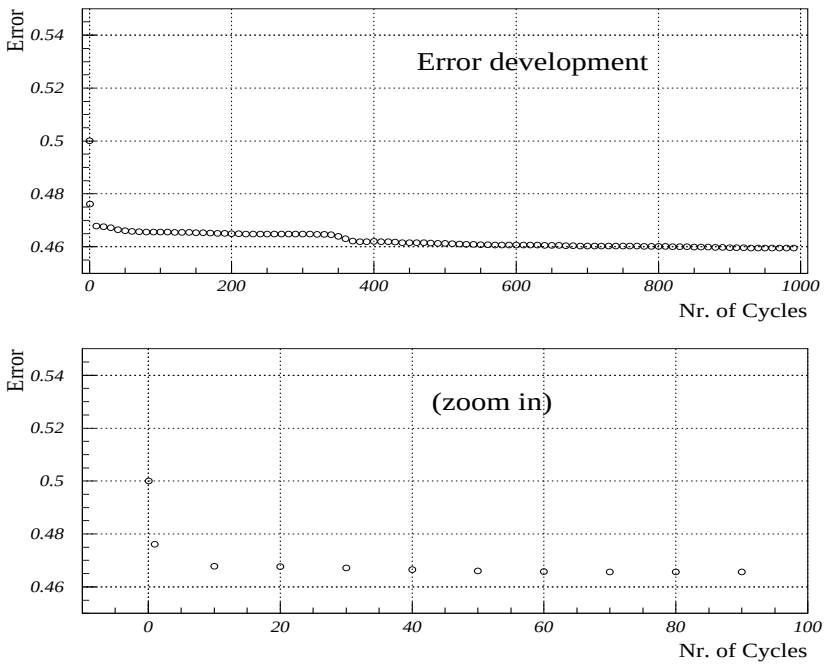


Figure 6.2: The average training error development during the 1000 cycles training of the 20-10-1 net.

Figure 6.2 shows the development of the average training error per epoch[1] for the first 1000 cycles of training, after that the average error changed very little. In section 6.3 the number of weight updates for 1000 training cycles was calculated to be 6.6 million, which should scale up to 99 million after 15000 cycles. Comparing with the DELPHI analysis [24], based on a 19-25-3 net, one finds that their figures are a factor 20 and 300 lower. But they do underline that previous studies have shown that nets trained with kinematical variables "takes considerably longer to train" than with eventshape variables. Thus one should expect to need many weight updates.

Figure 6.4 shows the purity and efficiency histograms for the two nets compared with the jetcharge. The nets are doing surprisingly well, though jetcharge is giving a better efficiency for low cut values. From figure 6.5 it is clear that in the purity-efficiency region where A_{max} is reached the jetcharge is superior to the neural nets. Results are summarized in tables 6.1 and 6.4.

[1] In `JetNet` the default value of one epoch is 1000 training patterns, so there are 66 or 69 epochs in one training cycle, depending on whether the hadronic cuts are used or not.

Method	A_{max}	Q_{cut}	$\rho(\epsilon = 1)$
1000 cycles 20-10-1 NN	0.2399±0.0029	0.24	0.6098±0.0017
15000 cycles 20-10-1 NN	0.2435±0.0033	0.17	0.6103±0.0017

Table 6.1: Results for the nets with q and p input variables

6.5 Adding p_t information

After a promising start, using p and q information as input, the next step was to add p_t. To limit the number of weights due to a new variable, p_t as defined in section 6.2.3, the number of hard tracks used as input was reduced.

In this thesis only one configuration with p, q and p_t information was tested, simply due to time limitations: It had 21 input neurons, matching 6 input tracks and one combined pseudotrack, from each hemisphere. The number of hidden neurons was increased to 25 to give the net more freedom to encode a possible solution of the $B - \bar{B}$ separation problem. This net was trained in two sessions, one for 1000 and another for 2000 cycles.

The purity versus efficiency plots in figure 6.6 show that the net, based only on p and q information, is just as good as the nets trained with the additional p_t information. In fact it is even better than the 21-25-1 net that trained only 1000 cycles, the reason for this may be that the 21-25-1 net have more degrees of freedom than the 20-10-1 and thus require more training to find a good weight configuration.

In table 6.2 the A_{max} values are listed, with corresponding cut values on the absolute netcharge. Comparison with table 6.1 shows that the 2000 cycles trained 21-25-1 net gives almost the same result as the 15000 cycles trained 20-10-1 net.

Method	A_{max}	Q_{cut}	$\rho(\epsilon = 1)$
1000 cycles 21-25-1 NN	0.2362±0.0029	0.25	0.6039±0.0017
2000 cycles 21-25-1 NN	0.2441±0.0032	0.17	0.6124±0.0017

Table 6.2: Results for nets with p_t information added

6.6 Adding beauty to the analysis

The last variable used as net-input in this analysis was the track probability from the B-tagging package, as described in section 6.2.4. This brought the total number of input variables up to four, thus increasing the size of the

nets even more. To organize all the tests made with this number of variables three structures can be singled out:

1. 28-35-1 structure with 6 hard tracks and one pseudotrack as input
2. 20-30-1 structure with 4 hard tracks and one pseudotrack as input
3. 32-45-1 structure with 7 hard tracks and one pseudotrack as input

Several versions of the 28-35-1 case were trained and tested. The hadronic cuts, below, refer to the selections listed in Sec 5.1. For simplicity four cases are defined. The update rules for α and η, the JetNet training parameters defined in section B.1, and the hadronic cuts were used unless something else is stated:

A: 28-35-1 net trained 2000 cycles without hadronic cuts imposed
B: 28-35-1 net trained 2000 cycles with constant α and η
C: 28-35-1 net trained 2000 cycles
D: 28-35-1 net trained 4000 cycles

By comparing A and C one should be able to see what kind of effects the hadronic cuts have on the nets, while case B versus C should reveal how the updating of the JetNet parameters α and η affects the performance of the nets. Finally, the C and D cases may point out if the nets were overtrained.

Upon testing the nets and comparing cases A, B and C with each other (figure 6.7) hardly any differences were found. The equal results in case A and C indicates that the hadronic cuts did not help the nets. A reason for this may be that the nets sort out most tracks, that would have been removed by the hadronic cuts, as noise. The equality between case B and C is more disturbing, at first glance this indicates that the updating rule for α and η have no effect on the net performance. The JetNet package is either so robust that the nets learn no matter what one do about the parameters, or else the given problem is so simple that the minimum of χ^2 is found without tuning the learning parameters.

When testing the neural nets in case C and D only minor differences were observed, most clearly in the efficiency plots shown in figure 6.8. A_{max} in case D turned out to be almost 1σ better than case C, see table 6.3, so the 28-35-1 net was certainly not overtrained.

After the 28-35-1 cases a structure with a higher ratio of hidden to input neurons was tested. For the 28-35-1 net this ratio was $\frac{35}{28} = 1.25$, a new ratio of 1.5 was tried instead. To reduce the size of the net, and thus the training time, the number of input neurons were reduced by using fewer input tracks from each hemisphere. The choice fell on a 20-30-1 structure. Again the hadronic cuts and updates of α and η were used, unless something else is stated.

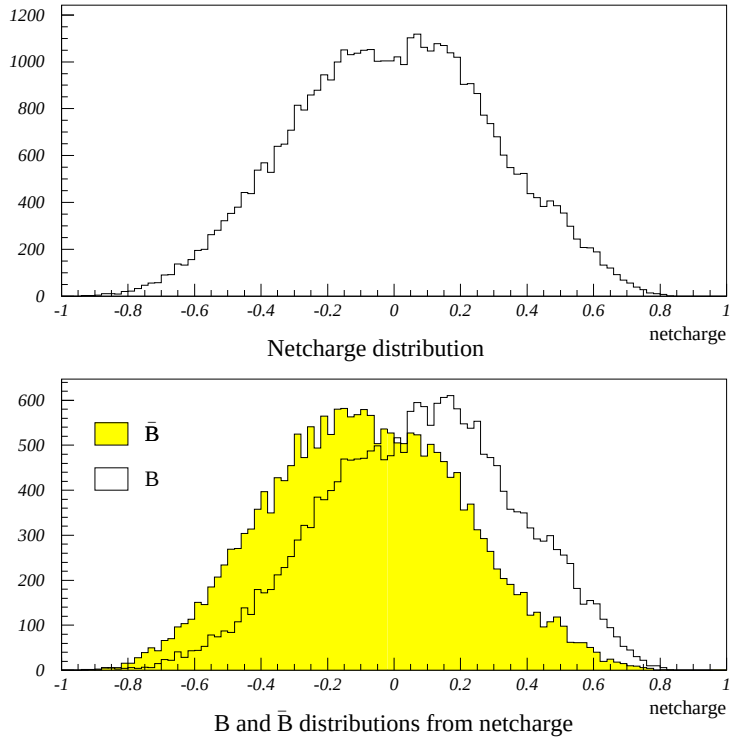


Figure 6.3: Netcharge distribution from a 32-45-1 net based on q, p, p_t and B-tag information. Should be compared with the Jetcharge plots in figure 6.1

Three cases of the 20-30-1 structure were trained and tested.

ξ: 20-30-1 net trained 1000 cycles

ν: 20-30-1 net trained 2000 cycles with constant α and η

ζ: 20-30-1 net trained 2000 cycles

In figure 6.10 the purity versus efficiency plots for these three cases are shown. It seems clear that case ξ was trained too short, it is significantly below the two others. But unlike the results for the 28-35-1 nets there was a positive effect from updating α and η, though not by much, as case ζ is only slightly better than case ν. Table 6.3 shows that case ζ gives the best value for A_{max}. Compared to the results from the 28-35-1 structure the 20-30-1 nets gave significantly better results, a 1.5σ higher A_{max} value in the best case.

After the good results with the small 20-30-1 structure it was decided to train a very large structure to see if more weights would make it easier for the

Method	A_{max}	Q_{cut}	$\rho(\epsilon = 1)$
case A	0.2437±0.0030	0.20	0.6106±0.0017
case B	0.2472±0.0032	0.17	0.6124±0.0017
case C	0.2447±0.0029	0.22	0.6108±0.0017
case D	0.2469±0.0029	0.23	0.6124±0.0017
case ξ	0.2390±0.0030	0.22	0.6065±0.0017
case ν	0.2498±0.0032	0.17	0.6143±0.0017
case ζ	0.2516±0.0032	0.17	0.6145±0.0017
case θ	0.2451±0.0030	0.20	0.6100±0.0017

Table 6.3: Results for 4 variable input nets

net to solve the $B - \bar{B}$ separation problem. A 32-45-1 structure was chosen, containing 1485 weights which is a factor 3 more than the promising 20-30-1 structure. The 32-45-1 net was trained 3000 cycles with the hadronic cuts on the data and updates of α and η enabled (case θ). In figure 6.11 the net is compared to the best 20-30-1 case, the 20-30-1 net is clearly better. By increasing the number of training cycles the 32-45-1 net may improve, but the cost of a much longer training time is probably higher than the gain from an improved result: Table 6.3 shows that A_{max} for the best 20-30-1 net (case ζ) is more than 2σ better.

Instead of trying out further structures a last attempt was made on the best case, the 20-30-1 net. It was trained for 3000 and 4000 cycles with the hadronic cuts and α and η updates enabled. The 3000 cycles version gave better results than case ζ, figure 6.12 shows how this net performs compared to the jetcharge and table 6.4 lists the results. The 4000 cycles version had a worse generalization performance than the 3000 cycles, it overtrained.

Method	A_{max}	Q_{cut}	$\rho(\epsilon = 1)$
Jetcharge algorithm	0.2567±0.0033	0.13	0.6185±0.0017
3000 cycles 20-30-1	0.2534±0.0031	0.19	0.6154±0.0017

Table 6.4: Results for the best 4 variable input net and the jetcharge

It is clear that the 20-30-1 structure is very close to the performance of the jetcharge in separating B and $\bar{B}$, but a nagging doubt was growing stronger: *Is the jetcharge an ultimate limit that the nets converge towards?* This question was not answered before the very late stages of this analysis, and will be addressed in chapter 7.

6.7 Plots from *The analysis, part 1*

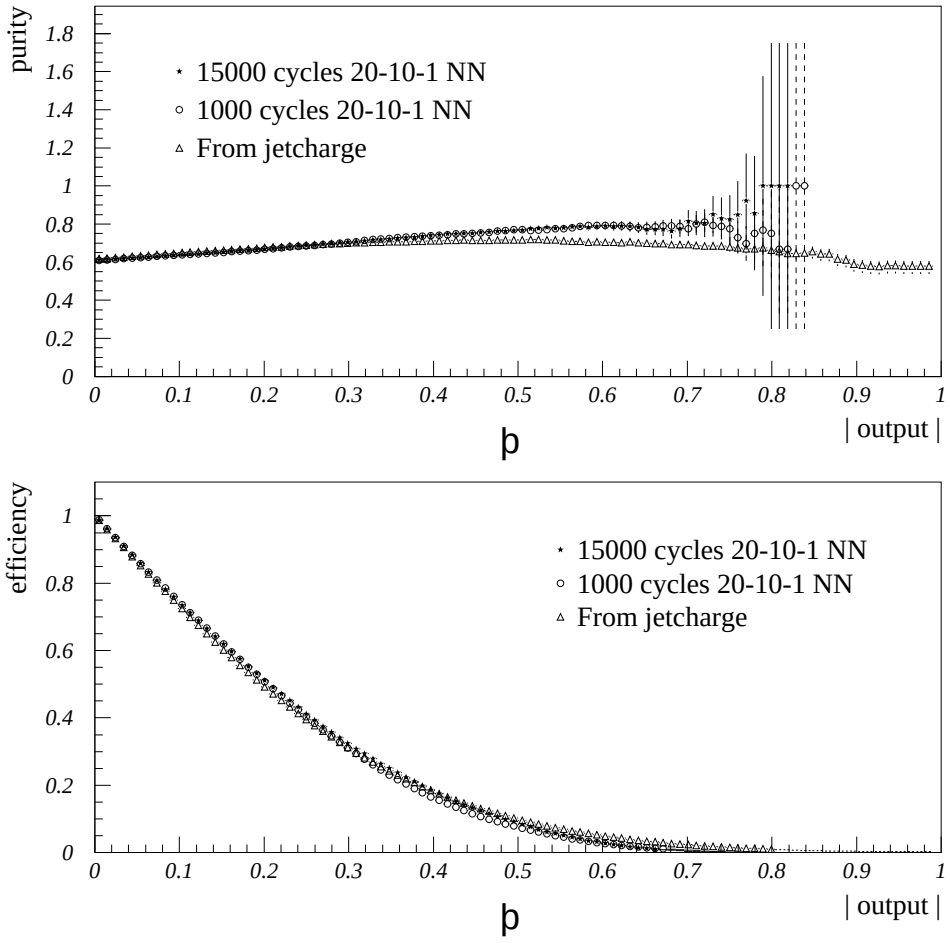


Figure 6.4: Comparing the tagging purity and efficiency for two nets, trained with p and q information, and the standard jetcharge. The purities and efficiencies are plotted as functions of cuts on the net-output (or jetcharge).

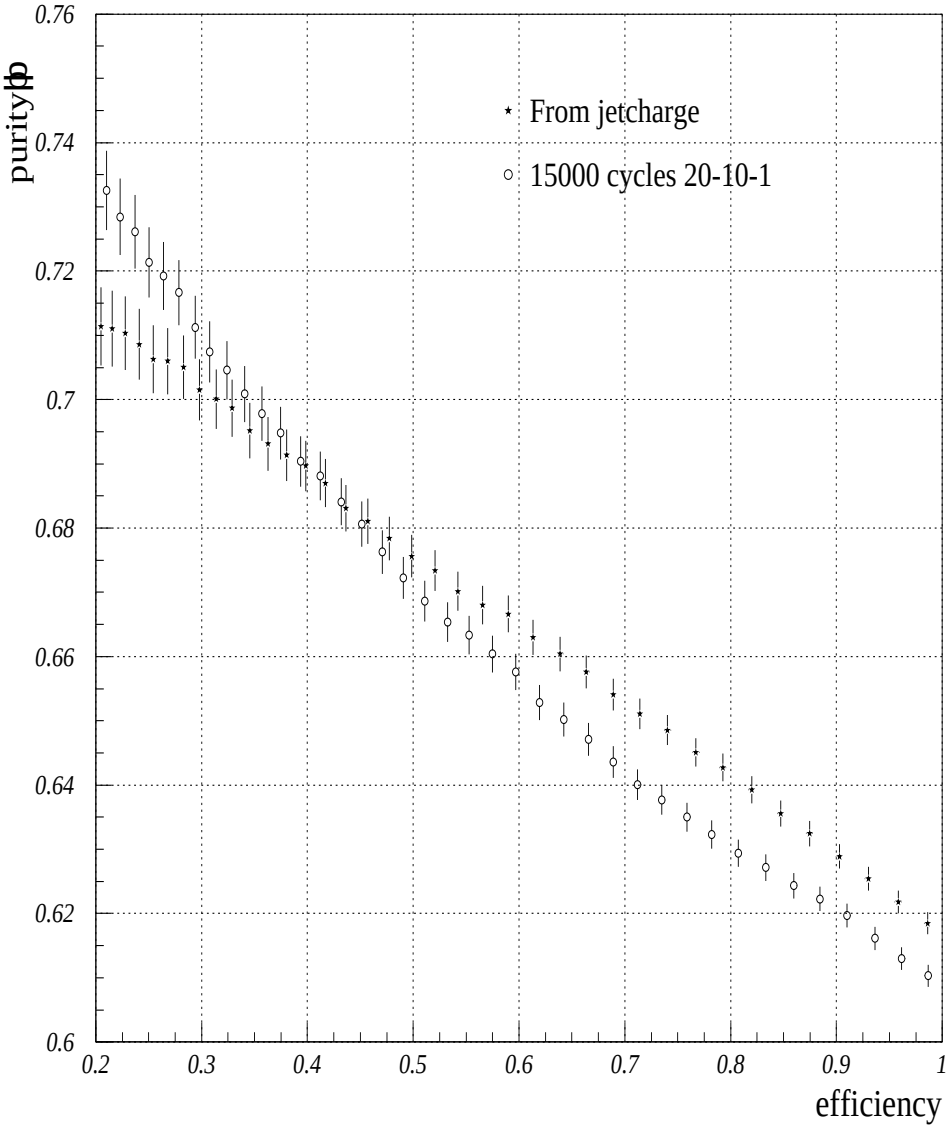


Figure 6.5: Comparing ϵ versus ρ plots for the 15000 cycles trained 20-10-1 net and for the standard jetcharge method

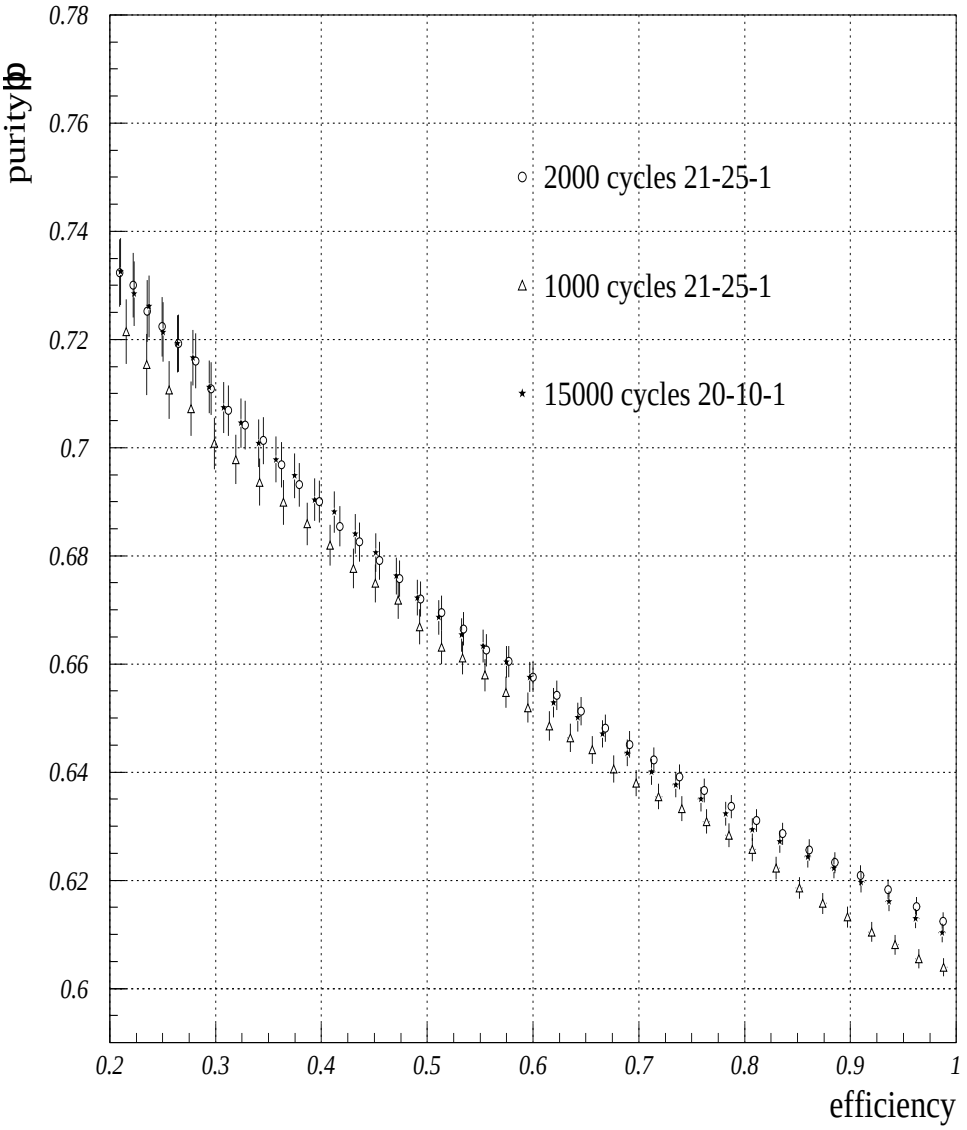


Figure 6.6: Comparing ϵ versus ρ plots for two 3 variable (P, Q, P_t) nets to that of a 2 variable (p and q)

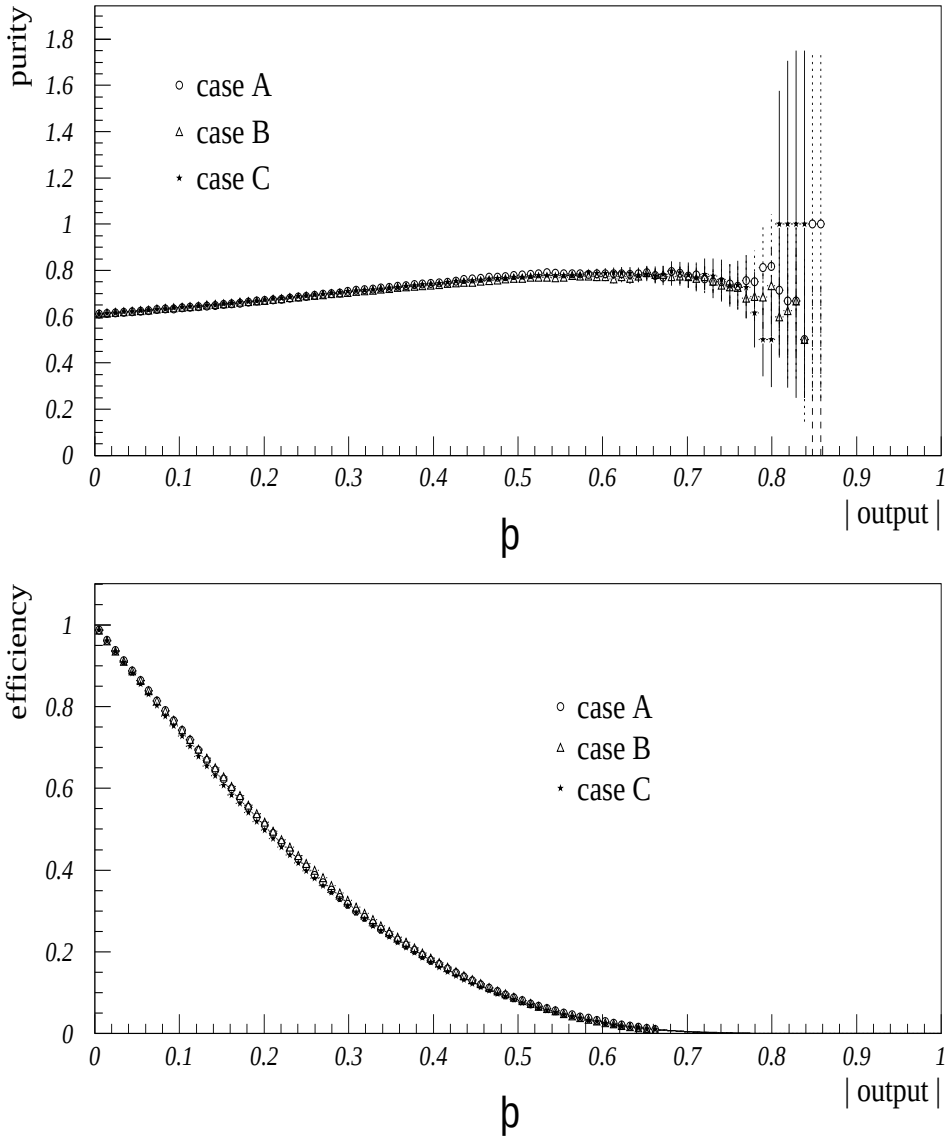


Figure 6.7: Comparing three 4 variable input 28-35-1 nets. Cases A, B and C were all trained 2000 cycles. The differences are that A was trained on data without hadronic cuts, B with constant α and η while in C both the hadronic cuts and updates of α and η were used.

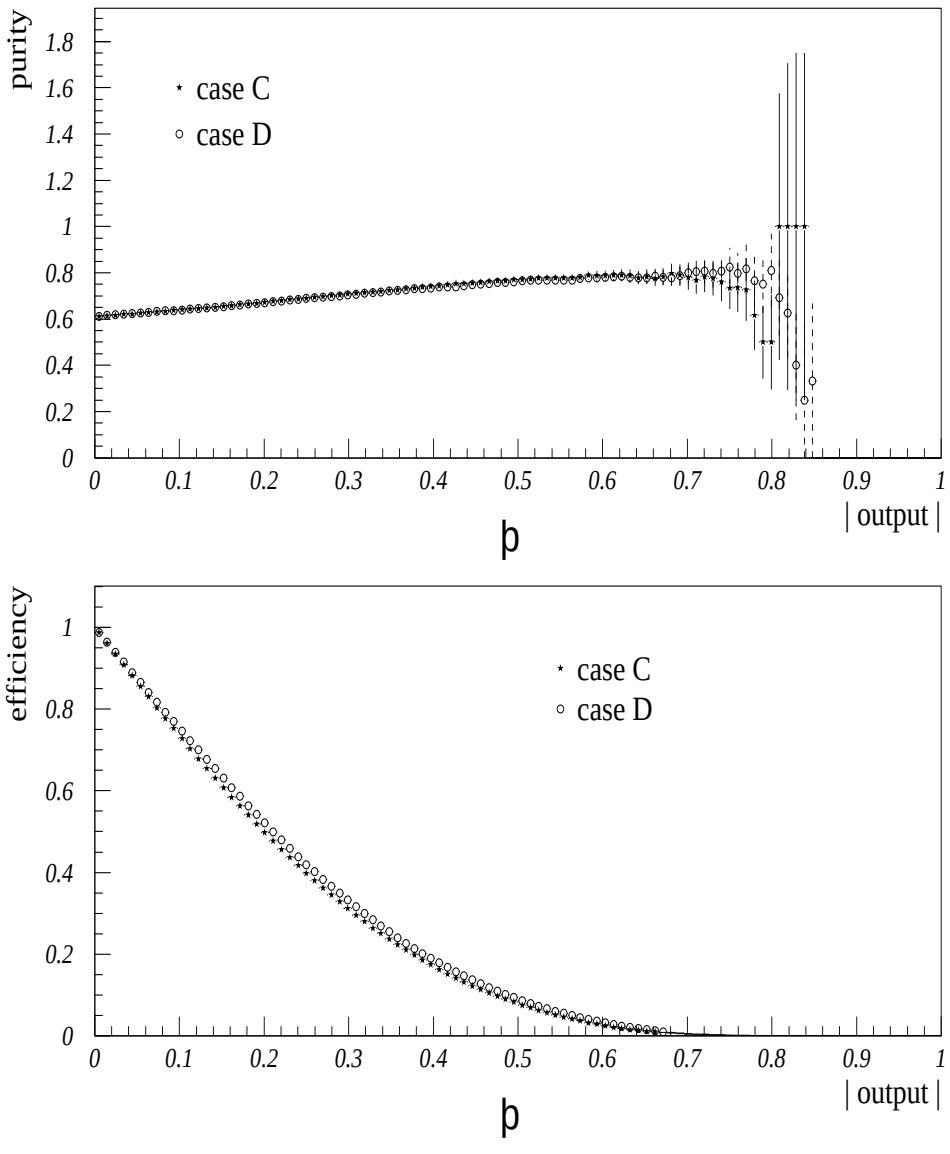
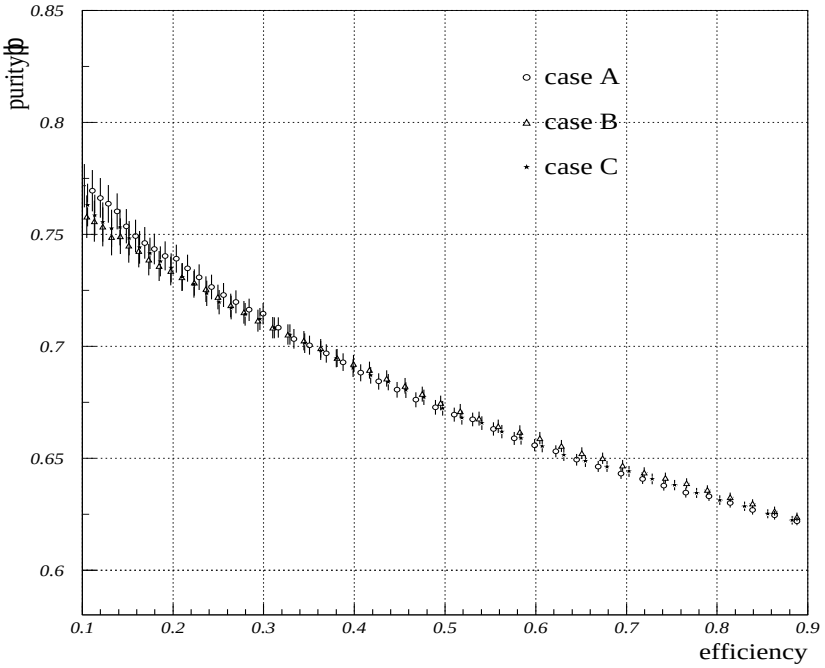


Figure 6.8: Comparing two 4 variable input 28-35-1 nets. Case C trained 2000 cycles and D 4000



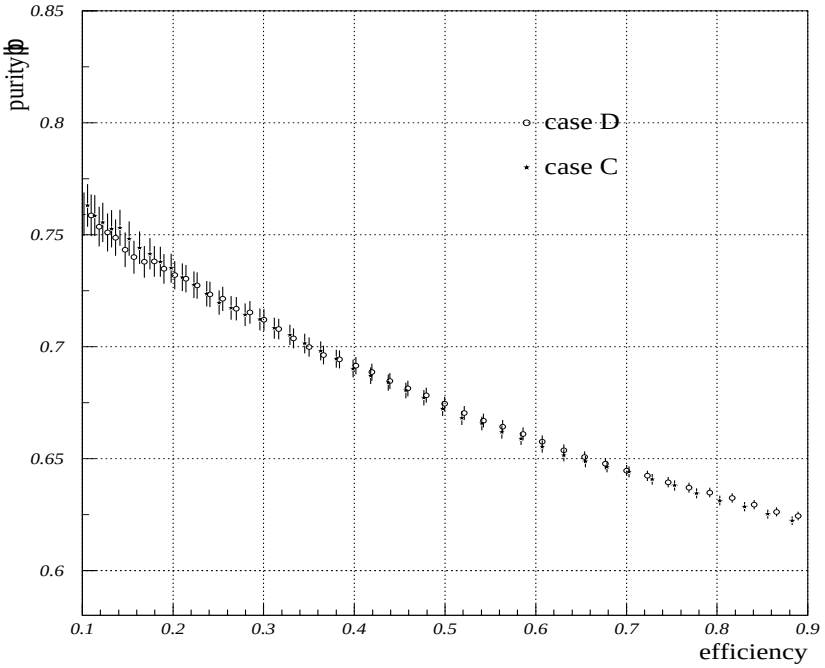


Figure 6.9: Comparing ρ versus ϵ for four 28-35-1 nets.

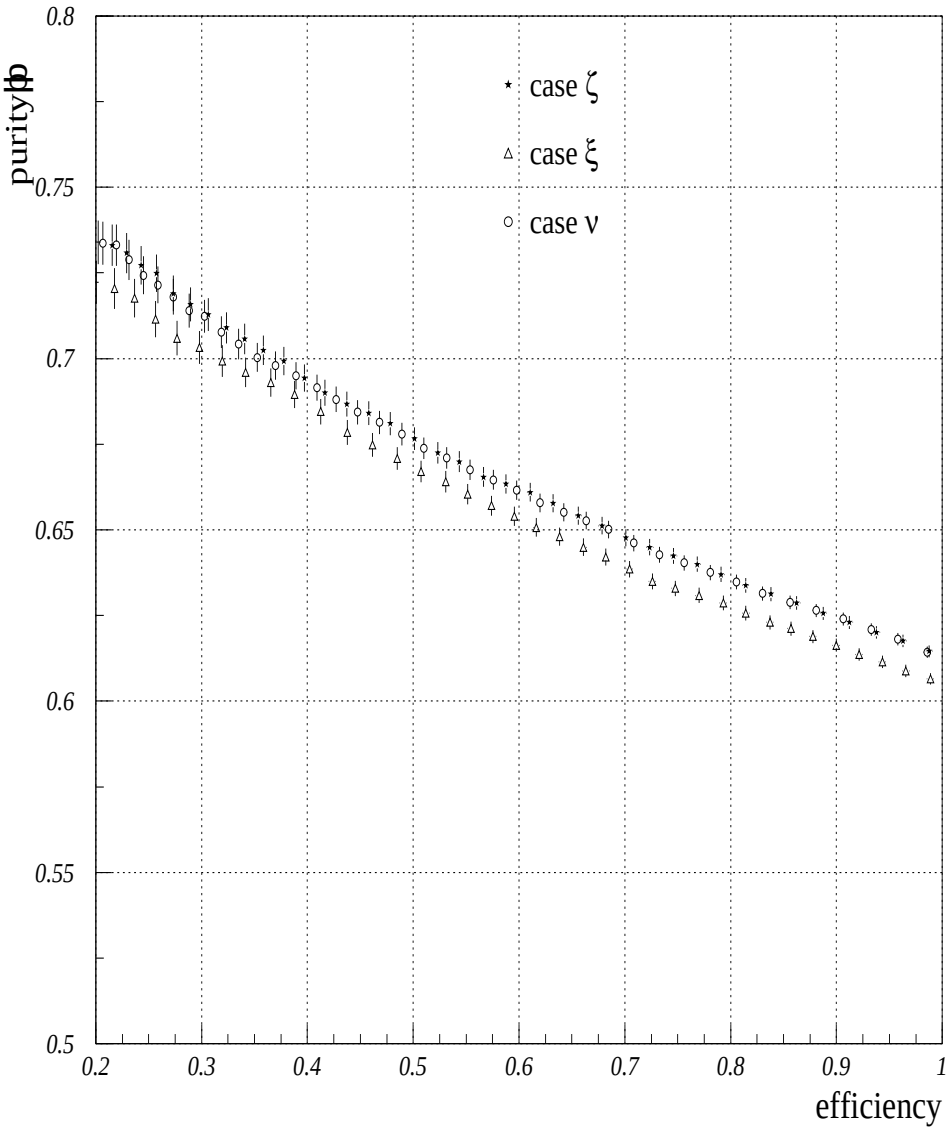


Figure 6.10: Comparing ϵ versus ρ plots for three 4 variable input 20-30-1 nets. Case ξ was trained 1000 cycles while cases ν and ζ were trained 2000 cycles, but in the case of ν with constant JetNet parameters α and η.

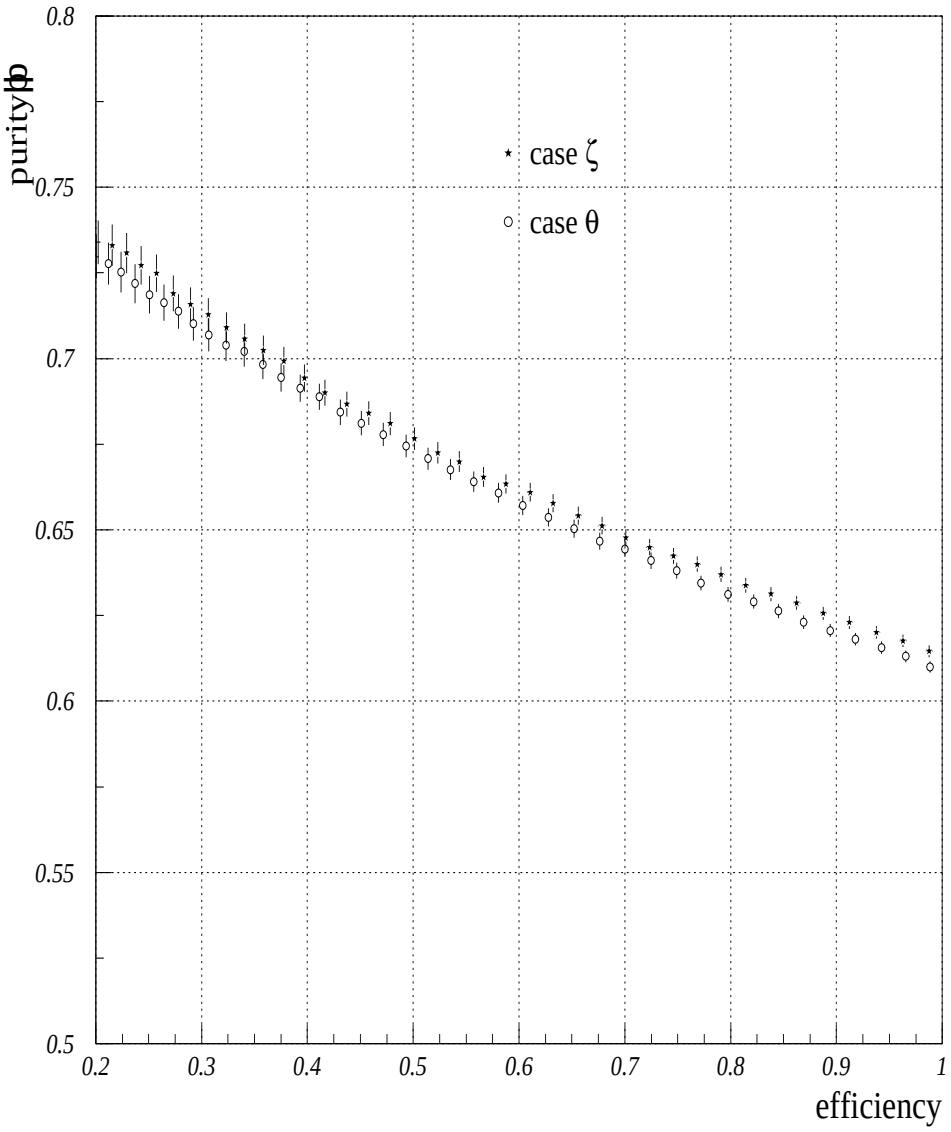


Figure 6.11: Comparing ϵ versus ρ plots for the two 4 variable input nets, one of structure 20-30-1 (case ζ) and one 32-45-1 (case θ). Both trained 2000 cycles with the hadronic cuts imposed on the data.

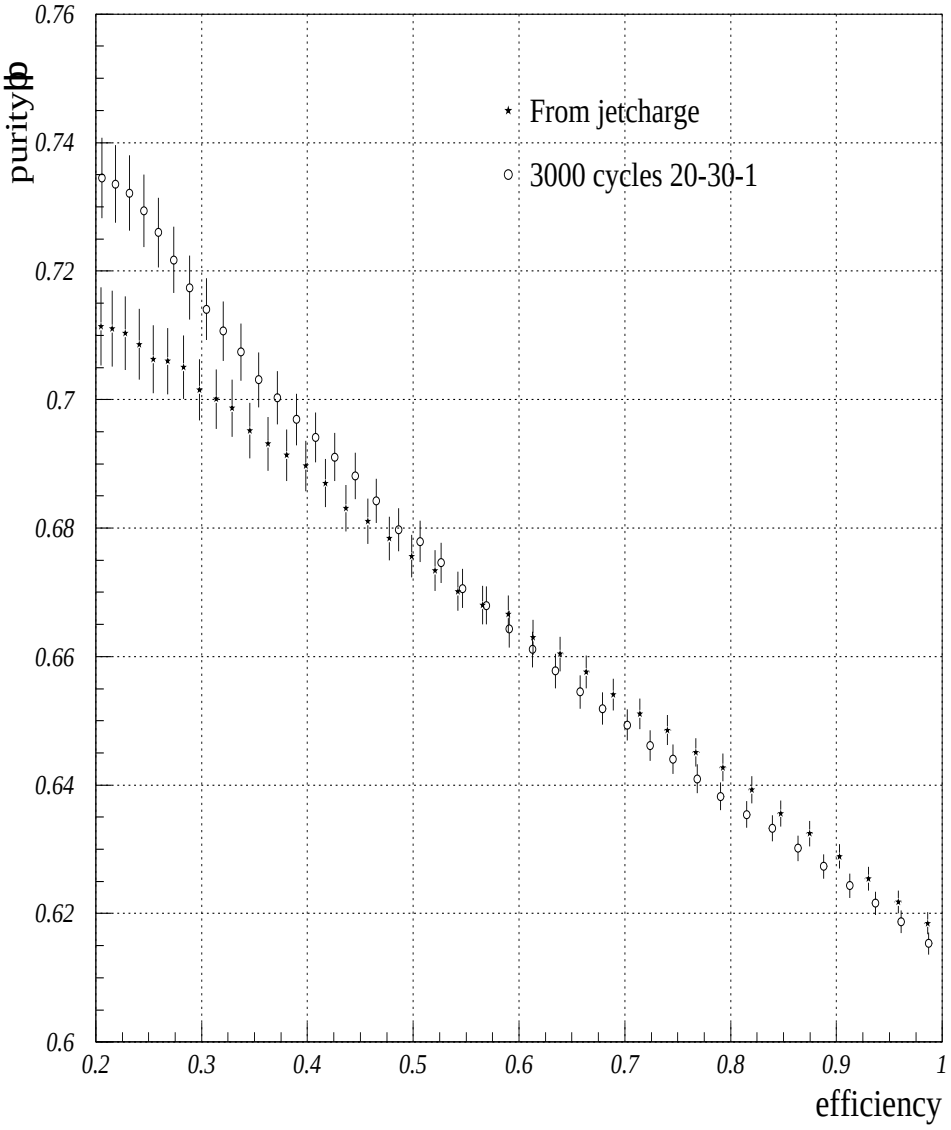


Figure 6.12: Comparing ϵ versus ρ plots for a 20-30-1 net trained 3000 cycles and the jetcharge.

Chapter 7

The analysis, part 2

In the previous chapter four basic input variables were explored, but no neural network configuration with a better performance than the jetcharge method was found. In this second part of the analysis a new approach is taken: Redefine the input variables[1].

7.1 Defining the new input variables

In chapter 6 the input variables p, p_t, q and $\log_b$ were used separately, the new approach is to weight p, p_t and $\log_b$ with q. Because $q \in \{-1, 1\}$ the new variables will be $\in [-1, 1]$, as long as they have been normalized. A natural first approach was to try $q{\cdot}p$, $q{\cdot}p_t$ and $q{\cdot}\log_b$ as input variables, where p, p_t and $\log_b$ are defined and normalized as in section 6.2.

Like in chapter 6 only a certain number of the hardest tracks were used directly to generate input variables from each hemisphere. If more tracks remained, after the n_{hard} tracks had been extracted, the information were combined into a pseudotrack. For the hard tracks the new variables could be used directly as input to the nets. For the pseudotrack a simple extension was tried: $\sum_i(q_i \cdot p_i)$, $\sum_i(q_i \cdot p_{t,i})$ and $\sum_i(q_i \cdot \log_{b,i})$ where the three sums run over the soft tracks.

These new definitions have reduced the number of input variables from each track, from 4 to 3, the number of input neurons will be reduced by the same factor. Hence the new variables have the positive side effect that they lower the number of connections in the net, which again should lower the training time.

In this part of the analysis the standard updates of η and α, as defined by Eqs (B.1) and (B.2), and the hadronic selection, as listed in section 5.1, were used permanently. Only a few net-structures were explored, due to the time constraints. Hence it is important to emphasize that the results in this chapter most likely can be optimized further.

[1]Suggested by Ole Røhne, at CERN, in June 1996

7.2 Never change a winning formula

From chapter 6 it is clear that the 20-30-1 structure, with only 4 hard tracks and one combined pseudotrack, from each hemisphere, is a good choice. Thus instead of starting from scratch, trying out new structures, that experience was used. The first attempt was made with a 15-30-1 structure, taking the same number of input tracks as the 20-30-1 net had done. This net was trained 2000 cycles, and the test showed a significant improvement compared to the nets trained with the old input variables. This is easily verified from figure 7.2, which show that the new net gives results up to 2σ better than the best net with the old input variables. The good results are quantified in table 7.1, A_{max} is almost 2σ better than that of the best 20-30-1 net listed in table 6.4.

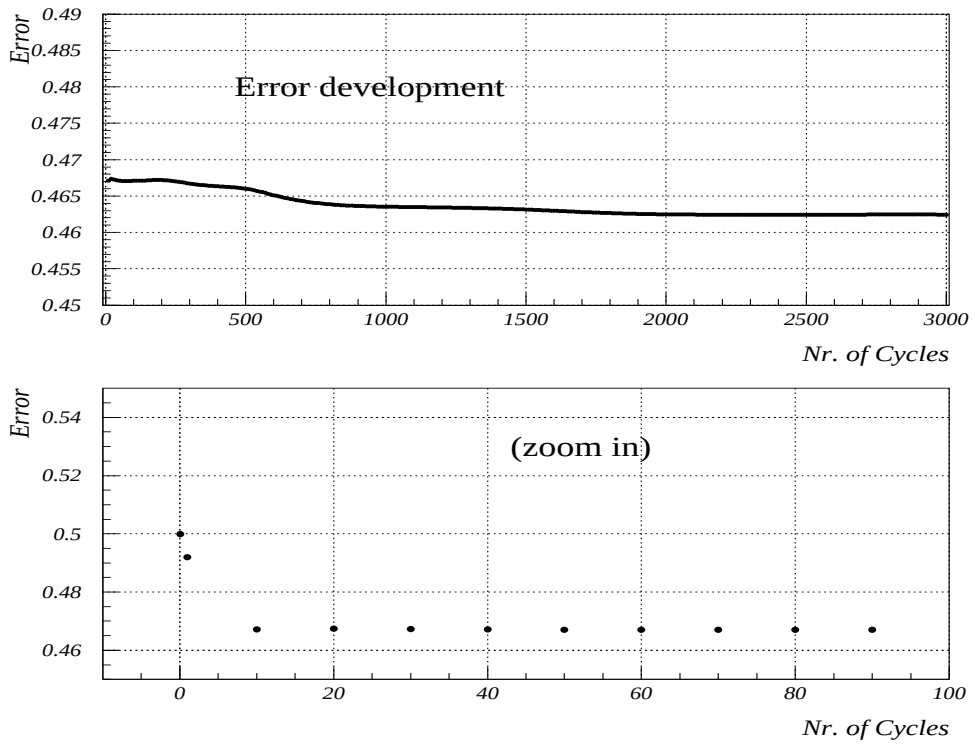


Figure 7.1: The average training error development during the 3000 cycles training of the 15-23-1 net.

With this good start it was clear that the new variables certainly made a huge difference. Two structures were briefly trained and tested with the three new input variables: 15-23-1 and 24-40-1.

The 15-23-1 structure was chosen because it has roughly the same ratio of hidden to input neurons as the good 20-30-1 net of the previous chapter. This structure was trained 1000, 2000 and 3000 cycles. As figure 7.1 shows the average training error decreased steadily the first 2000 cycles but then levelled out. Still the results did improve with the number of cycles, as can be seen from table 7.1, but only slightly. The A_{max} value hardly changed at all compared to the value for the 15-30-1 net.

In figures 7.3 and 7.4 the purity and efficiency plots based on the 3000 cycles trained net are compared to those from the standard jetcharge. The neural net is better than the jetcharge, by 1-2σ, for a large range of cuts on the output. This is even more visible in the ρ versus ϵ plot in figure 7.5. However in the interesting region for B-mixing, the A_{max} from the net is only 1σ higher than the jetcharge based value. Refer tables 6.4 and 7.1 for comparison.

The 24-40-1 structure was chosen to see if the addition of three more tracks, from each hemisphere, would make any difference. The 24 input neurons would then correspond to 7 hard tracks and one pseudotrack. With all the extra weights of the 24-40-1 structure, 1000 compared to the 552 of the 15-23-1 net, it seemed likely that this net would require at least twice as much training. 4000 and 6000 training cycles were used.

But as one can verify from table 7.1 this structure did not give as good results as the 15-23-1 structure. Like in chapter 6, the best results were obtained using only a few input tracks from each hemisphere. In addition the 6000 cycles trained 24-40-1 structure did *worse* than the 4000 cycles based version, caused by overtraining.

Method	A_{max}	Q_{cut}	$\rho(\epsilon = 1)$
2000 cycles 15-30-1	0.2595±0.0032	0.17	0.6160±0.0017
1000 cycles 15-23-1	0.2596±0.0031	0.19	0.6156±0.0017
2000 cycles 15-23-1	0.2599±0.0031	0.18	0.6171±0.0017
3000 cycles 15-23-1	0.2604±0.0032	0.17	0.6168±0.0017
4000 cycles 24-40-1	0.2591±0.0031	0.20	0.6160±0.0017
6000 cycles 24-40-1	0.2577±0.0030	0.21	0.6161±0.0017

Table 7.1: Results based on the $q{\cdot}p$, $q{\cdot}p_t$ and $q{\cdot}\log_b$ input variables.

7.3 Omitting the b-tag information

A final test was made by omitting the b-tag information to see what effect this would have on the performance of the nets. A 14-23-1 structure was chosen in order to be able to compare the results with those from the two structures in section 7.2. Using the input variables $q \cdot p$ and $q \cdot p_t$ three training sessions with the 14-23-1 structure were launched; lasting 1000, 2000 and 3000 cycles.

When testing the nets their performance turned out to be significantly worse than the nets with b-tag information. By comparing tables 7.1 and 7.2 it is clear that the nets without b-tag information give an A_{max} value about 2σ lower than those with b-tag information. But the 14-23-1 structure still performs better than the best net with the old input variables in chapter 6!

Method	A_{max}	Cut	$\rho(\epsilon = 1)$
1000 cycles 14-23-1	0.2549±0.0030	0.21	0.6145±0.0017
2000 cycles 14-23-1	0.2548±0.0030	0.21	0.6149±0.0017
3000 cycles 14-23-1	0.2534±0.0030	0.21	0.6152±0.0017

Table 7.2: Results based on the $q \cdot p$ and $q \cdot p_t$ input variables. No b-tag info.

An interesting thing one can read out of table 7.2 is that the 14-23-1 structure reached its best weight configuration already after 1000 training cycles. From that point and out the net started to specialize in remembering the training sample, thus degrading the generalization performance.

7.4 Plots from *The analysis, part 2*

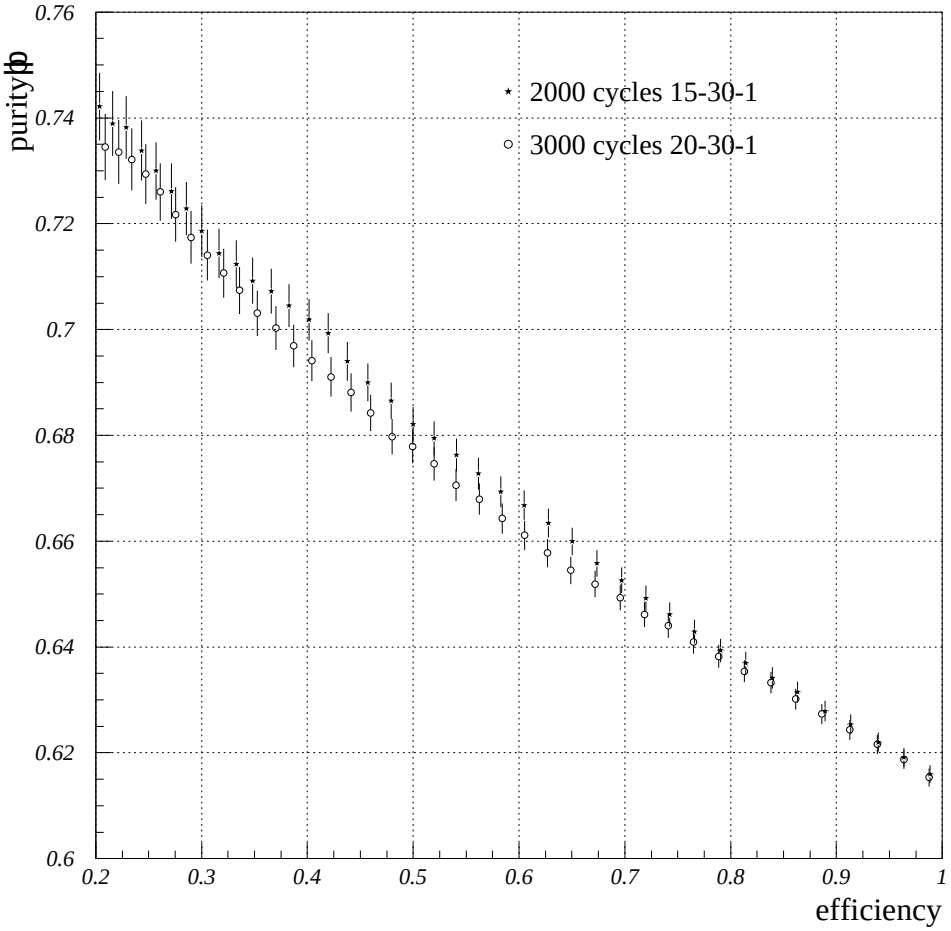


Figure 7.2: Purity versus efficiency plots for the best net with old input variables, 20-30-1, and the first net based on new input variables: 15-30-1. Both trained with the same number of input tracks from each hemisphere.

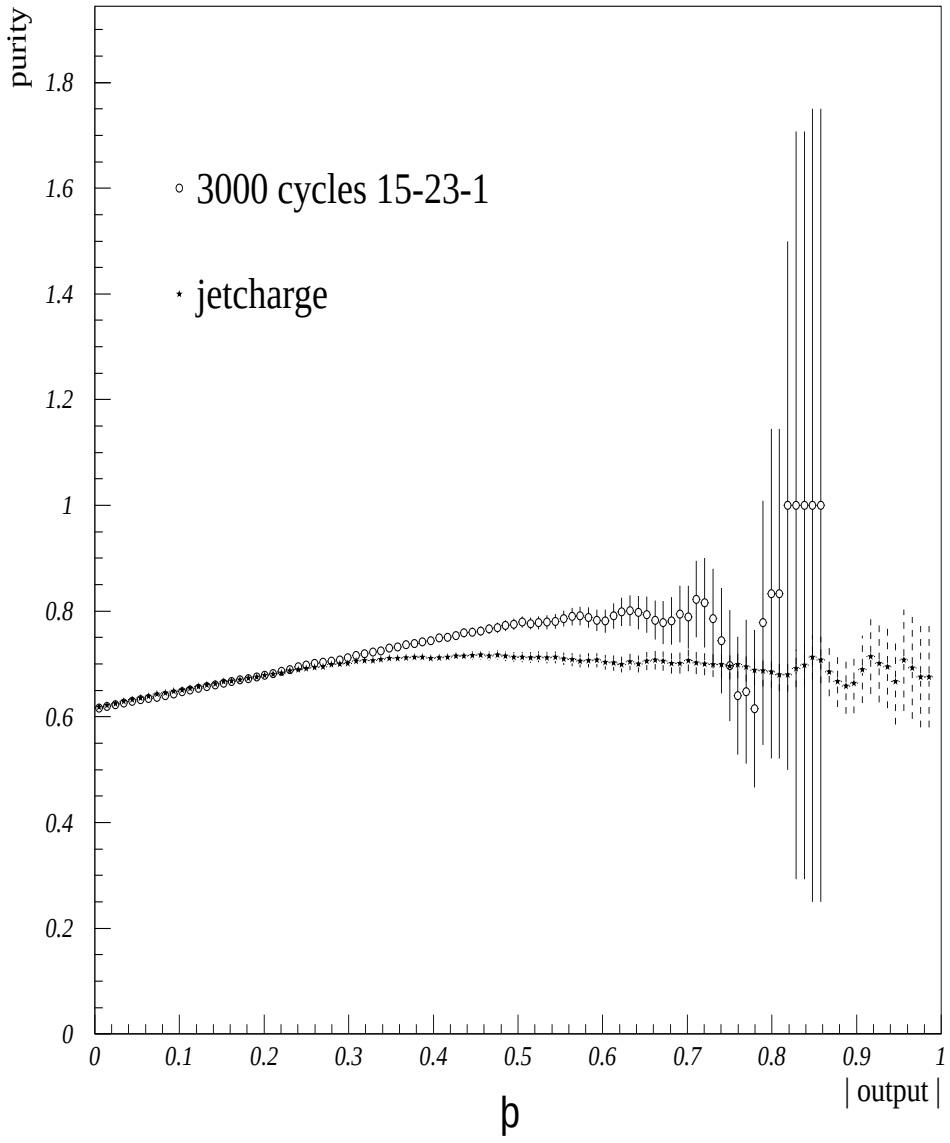


Figure 7.3: The purity plot for the best net with the new input variables, compared to the standard jetcharge result.

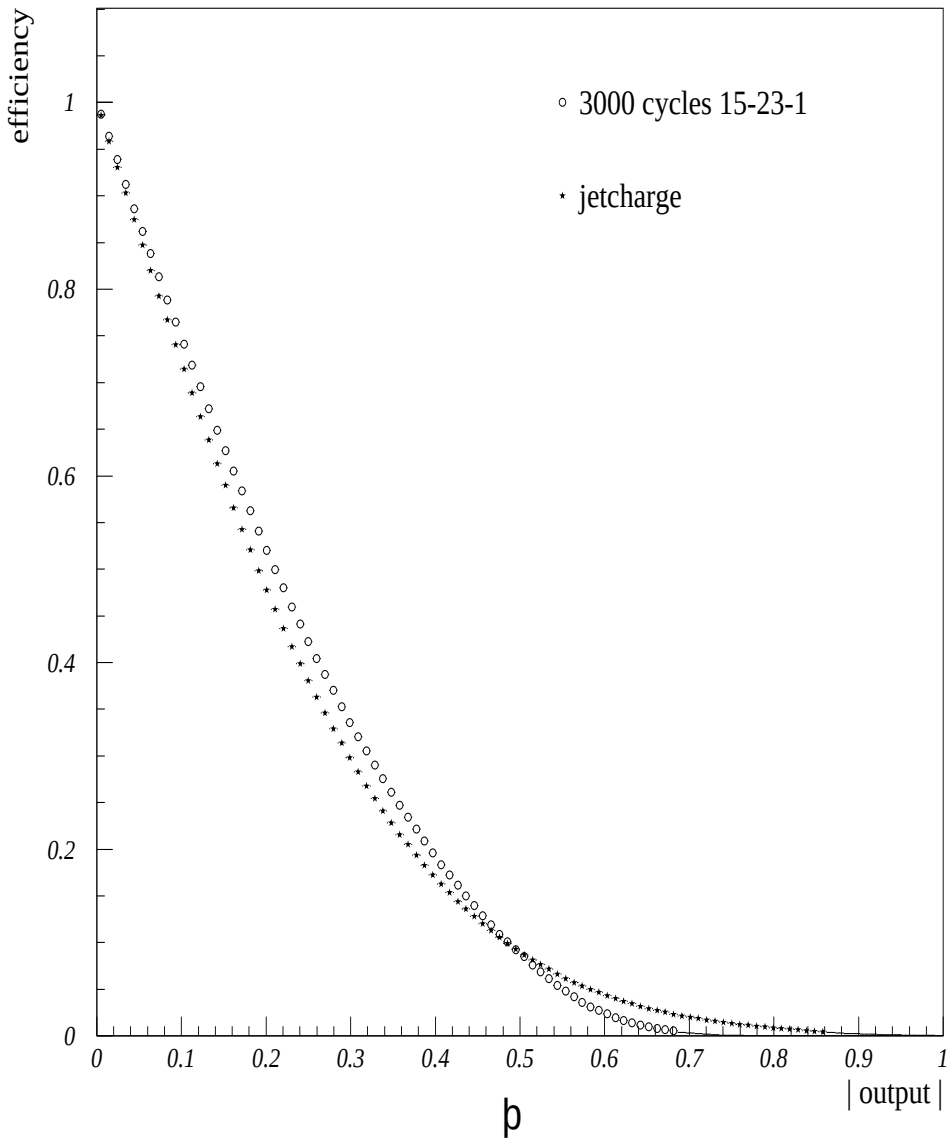


Figure 7.4: Efficiency plots for the best net with the new input variables, compared to the standard jetcharge result.

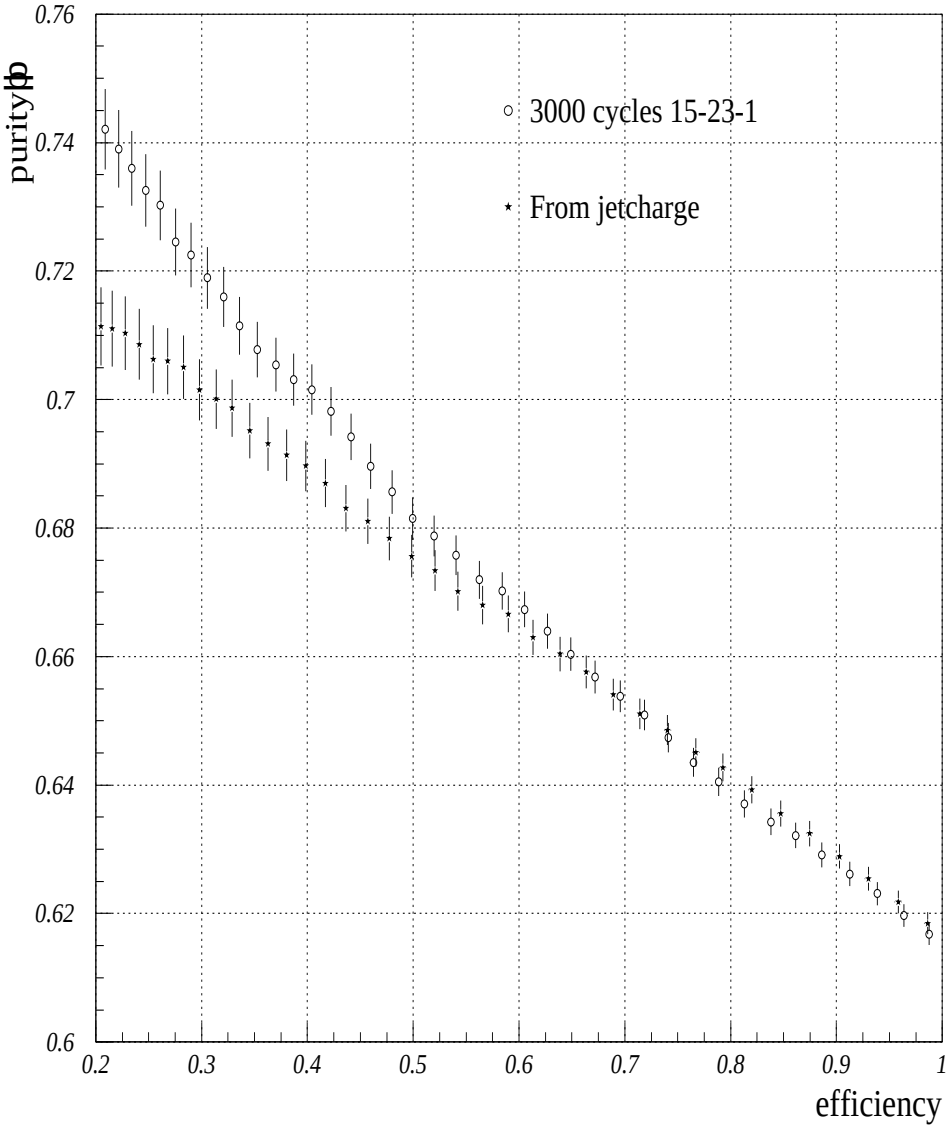


Figure 7.5: A purity versus efficiency plot for the best net with the new input variables, compared to the standard jetcharge result.

Chapter 8

Conclusions

The aim of this analysis was to train and test neural networks to see if they could perform the task of separating B and $\bar{B}$ patterns better than the Jetcharge method mentioned in section 3.3. After deciding what kind of neural net to use, the JetNet package of the feed-forward class, the analysis can roughly be split into three questions:

1. How does the number of hard tracks, few versus many, used as input to the net affect its performance?

2. Are the internal net-structure and parameters important when looking for the optimal solution?

3. Which input variables are the nets most sensitive to, and can different combinations of these affect the performance of the nets?

The three following sections will address these questions and try to answer them, based on the experience from chapters 6 and 7. In the last section, of this chapter, possible improvements to the given analysis will be discussed.

8.1 The number of hard tracks

How the number of hard tracks used as input, from each hemisphere, affected the nets was first tested in section 6.6. From tables 6.3 and 6.4 it is clear that the 20-30-1 net, with only 4 hard and one combined pseudotrack as input, performs best. However, the 32-45-1 net was not fully tested so a conclusive statement can not be drawn from those data. But with the new input variables in section 7.2 things looks more definite. From table 7.1 it is clear that the 15-23-1 net performs better than the 24-40-1 net, though the differences are below one σ.

Thus it is tempting to conclude that the number of hard tracks used as input to a net should be limited to 4 or 5, which also has the positive effect of reducing the training time and RAM-usage for the net. An explanation for

this result may be that the B information is carried by the hardest tracks, hence including many tracks will not help. An alternative explanation is related to the software set up: The input array use n_{hard} tracks from each hemisphere. If n_{hard} is close to the average charged track mulitplicity of B hemispheres [25], $\langle n_B \rangle \approx 12$, many entries in this array will be zero. Then the net will be trained with many "zero-tracks" which may have the effect on the net of pulling the B and $\bar{B}$ output distributions closer together. If this effect is significant it is obvious that n_{hard} must be set much smaller than $\langle n_B \rangle$, thus 4-6 seems reasonable.

8.2 The net-structure and parameters

With backup from the JetNet manual only one hidden layer was used, but one can question if this really was sufficient. Too few hidden layers and neurons would have given a poor generalization performance, the net would hardly have been able to recognize the training sample. But the "one hidden layer"-solution gave results better than the highly optimized Jetcharge method, so there certainly were enough hidden neurons to encode a good discrimination surface to separate the B and $\bar{B}$ patterns. In the other end of the spectrum too many neurons would have given the net problems with making up its mind. From the saturation measure, which can be read out from the `JetNet` program and used to stop the training, the different layers of weights seemed to have just enough freedom: The saturation usually went up by a factor 10 to 100 during training, this signals that the net is making up its mind. Thus it looks as if one hidden layer was sufficient for the problem in this analysis. Two hidden layers may have worked even better but only to the cost of a much longer training time.

Once the different nets had been thoroughly trained, and then been fed the testing sample, the differences in their A_{max} values seemed to depend more on the input variables than the exact net-structure. Well-trained nets using the old input variables, in chapter 6, gave A_{max} values within 1σ of each other. The same goes for well-trained nets using the new input variables, in chapter 7. But the A_{max} value of the best net using the new input variables was more than 2σ better than the best net using the old input variables, thus the number of hidden neurons and the exact structure seem to be less important than the input variable definitions. But even though the number of hidden neurons seem to be less important than other factors it looks as if the best results can be obtained using more hidden than input neurons.

But what about the net-parameters? In this analysis only α and η, the JetNet learning parameters, were tested. This took place in chapter 6. First for the 28-35-1 net, case B and C, and then for the 20-30-1 net, case ν and ζ. In all these cases the same number of training cycles were used, but B and

ν had constant values of α and η, while C and ζ used the update rules from section B.1. Then by comparing case B to C and case ν to ζ the effect from the learning parameters should be visible. Table 6.3 shows that case B, with constant α and η, is better than case C, with the updates. For the 20-30-1 net it is the other way around: Case ν, with constant α and η, performs worse than case ζ, with the updates. These results seem to contradict each other. But one should note that the results for case B and C are within 1σ of each other, and the same for case ν to ζ, so this could just as well be statistical fluctuations.

One explanation could have been that the 28-35-1 net has 1015 weights against the 630 of the 20-30-1 net and thus require more training before a good solution is reached. But even after 4000 cycles, with updates of α and η, labeled case D, the version with constant parameters throughout the training performs better.

There are two possible conclusions to this section. Either the `JetNet 3.0` package is so robust and well-written that the default values of the parameters give fairly good results. Or else the problem of separating B and $\bar{B}$, with the given set of variables, is so simple that no fine-tuning of the learning parameters are needed. Even though no single conclusion can be based on these results the updates of α and η were used, per default, in chapter 7 since the best net in chapter 6 was obtained with these updates enabled.

8.3 The input variables

In chapter 6 four input variables, q, p, p_t and $\log_b$, were defined and used to train nets in separating B and $\bar{B}$ patterns. In chapter 7 these variables were recombined into $q{\cdot}p$, $q{\cdot}p_t$ and $q{\cdot}\log_b$ and then used as input to the nets. Though no new information was added in chapter 7 the significantly better results, more than 2σ, indicate that the nets are much more sensitive to these variables than those defined in chapter 6.

The test done by removing the b-tag information from the new set of input variables, in section 7.3, clearly indicates that the b-tag information is important for the nets in order to separate the B and $\bar{B}$ patterns.

If one compares the variation in the performance of the nets, due to changes in the input variable definitions and due to changes in the net-structures or parameters, the conclusion must be that the input variables are most important. Thus future improvements will most likely be done by even more sensitive definitions of the input variables (and by adding new information as well, of course). But to obtain optimal solutions one will also have to adjust the net-structure and parameters.

8.4 General comments and reflections

There are many effects one must take into consideration when looking for solid conclusions in this analysis, but there is in particular one conclusion that the data strongly supports: The amount of required training.

In an older DELPHI analysis [24] it was noted that "a network based on kinematical input variables could perform slightly better than a network using eventshape variables, but takes considerably longer to train". Indeed, while the DELPHI analysis needed 300 thousand updates of the weights the current analysis never used less than 1000 training cycles, or 6.6 million weight updates. And it is evident from the tables in chapter 6 and 7 that in most cases the neural nets would improve further even after the initial 6.6 million weight updates. Thus *networks based on kinematical variables do require more training.*

One can take a closer look at how the learning took place by studying the training error development. Strictly speaking plots of the average training error, as a function of the number of training cycles, do not reveal how the generalization performance develops. But they do show the activity of the weight changes. Thus when the average training error levels out the weight changes in the network is very small, which signals that the main part of the learning process is over.

For the simple 20-10-1 net with only two input variables, refer figure 6.2, most of the learning took place up to 400 cycles. After that point only minor changes occured. For the more complex nets, with more weights and input variables, it took more training for the nets to sort out the specific B and $\bar{B}$ features. The 15-23-1 net, trained on the new input variables, learned most of these features up 700-800 cycles. But as one can see from figure 7.1 the net continued to learn even up to 2000 cycles, where the average training error leveled out.

8.5 Final comments

Based on the results from chapters 6 and 7 it has been underlined that the performance of the nets depend strongly on how the input variables are defined, as compared to the dependency on the number of input tracks and the specific net-structure and parameters. Thus a future development of this analysis should search for additional information or more sensitive input variable definitions.

- Additional information may include: Kaon, muon and lepton tags and possibly eventshape variables

- The nets were more sensitive to $(q \cdot p)$ than q and p, but maybe even better definitions of the input variables can be found: Like a jetcharge inspired $(q \cdot p)^\kappa$, $(q \cdot p_t)^\kappa$, $(q \cdot \log_b)^\kappa$ etc.

To find optimal solutions one would naturally have to continue with the variation of the net-structures and number of input variables, and also look closer on the use of the net-parameters. But this should come second to the exploration of the input variables.

However, the `JetNet 3.0` package seem to have fairly good default values, so by using between 5 and 7 input tracks from each hemisphere and roughly 50% more hidden than input neurons one should get a good start. The best net found in this analysis was a 15-23-1 structure trained and tested with the input variable definitions from section 7.1. It was trained 3000 cycles with hadronic cuts imposed on the data sample and updates on α and η, as defined in section B.1. The A_{max} value calculated from its B and $\bar{B}$ output was 1σ higher than the A_{max} value calculated from the jetcharge output, though for a higher Q_{cut}. But it has a purity at 100% efficiency about one σ lower than that from the jetcharge. Refer figure 7.5.

Method	A_{max}	Q_{cut}	$\rho(\epsilon = 1)$
jetcharge algorithm	0.2567±0.0033	0.13	0.6185±0.0017
3000 cycles 15-23-1	0.2604±0.0032	0.17	0.6168±0.0017

Table 8.1: The best net compared with jetcharge

The main conclusion of this thesis is that the aim of finding a neural network configuration that could perform the task of separating B and $\bar{B}$ hemispheres better than the Jetcharge method has been achieved. But the difference is still too small to be useful in a B_s^0 mixing analysis, thus further improvements are needed.

Appendix A

Statistics

Σ_b and Σ_{bb} denotes the integrated B and $\bar{B}$ distributions, from section 5.3.2. The aim of this appendix is to outline the errors in Σ_b and Σ_{bb} and other quantities, from chapter 5, that are based on the integrated distributions.

A.1 Errors in Σ_b and Σ_{bb}

As figure A.1 shows the integration goes from -1 up to a certain limit on the netcharge (or jetcharge). Let this limit correspond to bin q in the given distribution, then one can label the integrated value $n(q)$. This would then correspond to $\Sigma_b(q)$ in a B distribution or $\Sigma_{bb}(q)$ if it was a $\bar{B}$ distribution. Let N be the total number of patterns in the given distribution.

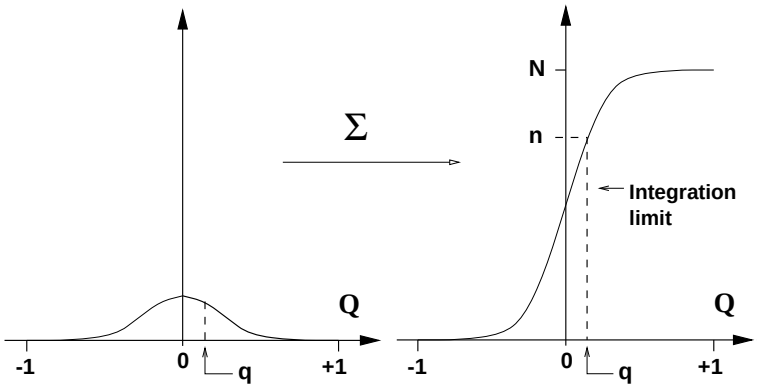


Figure A.1: Each distribution, both B and $\bar{B}$, is integrated from -1 to $+1$.

The error in $n(q)$ is defined as the square root of the variance given by Binomial statistics, thus the first step is to find an expression for the variance in $n(q)$.

From elementary statistics the following definition becomes useful [29]:

> Let $n_1, n_2, \cdots, n_k$ be a random sample from f_n(n; p_1,p_2,$\cdots$,p_k). Then an estimator $\hat{p}_q$=$g(n_q)$ is said to be unbiased for p_q if the expectation value $\langle \hat{p}_q \rangle = p_q$ for q=1,$\cdots$,k

In this definition the subscripts correspond to the bin numbers, $n_q = n(q)$, from the integrated distributions. Because n_q follows Binomial statistics its expectation value should be $\langle n_q \rangle$=$N \cdot p_q$ where p_q is the probability of integrating an event up to bin q. From this statement and the above definition one can easily show that $\hat{p}_q = \frac{n_q}{N}$ is an unbiased estimator for p_q:

$$\langle \hat{p}_q \rangle = \langle \frac{n_q}{N} \rangle = \frac{1}{N} \cdot \langle n_q \rangle = p_q \tag{A.1}$$

A random distributed variable in Binomial statistics will have a variance given by $\sigma^2 = N \cdot p \cdot (1-p)$ where p is the probability of success and N the number of trials. From this result an estimator for the variation in n_q can be given as:

$$\sigma^2_{\hat{n}} = N \cdot \hat{p}_q \cdot (1 - \hat{p}_q) \tag{A.2}$$

However, from Eq A.1 it is clear that $\frac{n_q}{N}$ is a good estimator for $\hat{p}_q$ so this can be substituted into Eq A.2 giving:

$$\sigma^2_{\hat{n}} = N \cdot \frac{n_q}{N} \cdot (1 - \frac{n_q}{N})$$

$$\Downarrow$$

$$\sigma_{\hat{n}} = \sqrt{\frac{n_q \cdot [N - n_q]}{N}}$$

By using $n(q)$ instead of n_q and renaming $\sigma_{\hat{n}}$ to $\sigma(q)$, just to emphasize its dependence on the cut bin q, the error in the integrated distributions can be written as

$$\sigma(q) = \sqrt{\frac{n(q) \cdot [N - n(q)]}{N}} \tag{A.3}$$

Eq (A.3) is valid for both B and $\bar{B}$ distributions as long as $n(q)$ gives the integrated number up to bin q and N the total number of the given B-flavor.

A.2 Error in the mean tagging efficiency ϵ

In section 5.3.4 the mean tagging efficiency ϵ was defined using the integrated $B - \bar{B}$ information in Σ_b and Σ_{bb}. To calculate the error in ϵ[1], based on the errors in Σ_b and Σ_{bb}, a useful theorem can be listed [30]:

[1]There is an alternative, and much quicker way of, of obtaining an error measure for ϵ. One simply forget about the $B - \bar{B}$ nature of the events and instead just counts the n number of successes. Since n=$\epsilon \cdot N$ follow Binomial statistics $\hat{\epsilon}$=$\frac{n}{N}$ and $\sigma_{\hat{\epsilon}}$=$\sqrt{\epsilon \cdot (1-\epsilon)/N}$.

Given a function $R = R(x_1, x_2, \cdots, x_n)$, with a known error s_i in each of its x_i variables, the error in R can be expressed as

$$s_R = \sqrt{\left(\frac{\partial R}{\partial x_1} \cdot s_1\right)^2 + \cdots + \left(\frac{\partial R}{\partial x_i} \cdot s_i\right)^2 + \cdots + \left(\frac{\partial R}{\partial x_n} \cdot s_n\right)^2} \qquad \text{(A.4)}$$

R and s_R can be identified as ϵ and σ_ϵ in the current problem. In section 5.3.4 ϵ was written as a function of $\Sigma_b(\beta_{lo})$, $\Sigma_{bb}(\beta_{lo})$, $\Sigma_b(\beta_{hi-1})$ and $\Sigma_{bb}(\beta_{hi-1})$. These four variables will contribute to the total error and must correspond to the x_i variables in Eq (A.4).

To reduce the complexity of the calculations the error contributions due to B and $\bar{B}$ terms are treated separately. In one part errors due to Σ_{bb}:

$$\chi^{bb}(q) = \left(\frac{\partial \epsilon(q)}{\partial \Sigma_{bb}(\beta_{lo})} \cdot \sigma_{bb}(\beta_{lo})\right)^2 + \left(\frac{\partial \epsilon(q)}{\partial \Sigma_{bb}(\beta_{hi} - 1)} \cdot \sigma_{bb}(\beta_{hi} - 1)\right)^2 \qquad \text{(A.5)}$$

And in the other part errors due to Σ_b:

$$\chi^{b}(q) = \left(\frac{\partial \epsilon(q)}{\partial \Sigma_{b}(\beta_{lo})} \cdot \sigma_{b}(\beta_{lo})\right)^2 + \left(\frac{\partial \epsilon(q)}{\partial \Sigma_{b}(\beta_{hi} - 1)} \cdot \sigma_{b}(\beta_{hi} - 1)\right)^2 \qquad \text{(A.6)}$$

Eqs (A.5) and (A.6) contain all the partial derivatives that can be calculated from ϵ, so in the spirit of Eq (A.4) one can set up the following formula for the total error in ϵ:

$$\sigma_\epsilon(q) = \sqrt{\chi^b(q) + \chi^{bb}(q)} \qquad \text{(A.7)}$$

To calculate the partial derivatives of Eq (A.5) the expression for ϵ given in Eq (5.2), in section 5.3.4, was used. It gives

$$\chi^{bb}(q) = \left(\frac{1}{N} \cdot \sigma_{bb}(\beta_{lo})\right)^2 + \left(\frac{-1}{N} \cdot \sigma_{bb}(\beta_{hi} - 1)\right)^2$$

$$\Updownarrow$$

$$\chi^{bb}(q) = \frac{1}{N^2} \cdot [\sigma_{bb}(\beta_{lo})^2 + \sigma_{bb}(\beta_{hi} - 1)^2] \qquad \text{(A.8)}$$

Similarly for Eq (A.6) one can obtain:

$$\chi^{b}(q) = \left(\frac{1}{N} \cdot \sigma_{b}(\beta_{lo})\right)^2 + \left(\frac{-1}{N} \cdot \sigma_{b}(\beta_{hi} - 1)\right)^2$$

$$\Updownarrow$$

$$\chi^{b}(q) = \frac{1}{N^2} \cdot [\sigma_{b}(\beta_{lo})^2 + \sigma_{b}(\beta_{hi} - 1)^2] \qquad \text{(A.9)}$$

Using Eqs (A.8) and (A.9) in Eq (A.7) the error in $\epsilon(q)$ can be written:

$$\sigma_\epsilon(q) = \frac{1}{N}\sqrt{\sigma_b(\beta_{lo})^2 + \sigma_{bb}(\beta_{lo})^2 + \sigma_b(\beta_{hi} - 1)^2 + \sigma_{bb}(\beta_{hi} - 1)^2} \qquad \text{(A.10)}$$

A.3 Error in the mean tagging purity ρ

The mean tagging purity $\rho(q)$ was defined in section 5.3.5. To calculate the error in this quantity the same approach as for $\epsilon(q)$, in section A.2, was used. The only difference is that every occurrence of ϵ must be replaced by ρ. From Eq (A.5) with $\epsilon \rightarrow \rho$:

$$\chi^{bb}(q) = \left(\frac{\partial\rho(q)}{\partial\Sigma_{bb}(\beta_{lo})} \cdot \sigma_{bb}(\beta_{lo})\right)^2 + \left(\frac{\partial\rho(q)}{\partial\Sigma_{bb}(\beta_{hi}-1)} \cdot \sigma_{bb}(\beta_{hi}-1)\right)^2$$

$$\Updownarrow$$

$$\chi^{bb}(q) = \left(\frac{\partial\rho(q)}{\partial\Sigma_{ri}(q)} \cdot \frac{\partial\Sigma_{ri}(q)}{\partial\Sigma_{bb}(\beta_{lo})} \cdot \sigma_{bb}(\beta_{lo})\right)^2$$

$$+ \left(\frac{\partial\rho(q)}{\partial\Sigma_{wr}(q)} \cdot \frac{\partial\Sigma_{wr}(q)}{\partial\Sigma_{bb}(\beta_{hi}-1)} \cdot \sigma_{bb}(\beta_{hi}-1)\right)^2 \qquad \text{(A.11)}$$

Next step is to use equations (5.4), (5.5) and (5.6) in Eq (A.11):

$$\chi^{bb}(q) = \left(\frac{\Sigma_{wr}(q)}{N^2} \cdot 1 \cdot \sigma_{bb}(\beta_{lo})\right)^2 + \left(\frac{-\Sigma_{ri}(q)}{N^2} \cdot (-1) \cdot \sigma_{bb}(\beta_{hi}-1)\right)^2$$

$$\Updownarrow$$

$$\chi^{bb}(q) = \frac{1}{N^4} \cdot \left([\Sigma_{wr}(q) \cdot \sigma_{bb}(\beta_{lo})]^2 + [\Sigma_{ri}(q) \cdot \sigma_{bb}(\beta_{hi}-1)]^2\right) \qquad \text{(A.12)}$$

Where $N = \Sigma_{ri}(q) + \Sigma_{wr}(q)$ is the number of classified patterns. With Eq (A.12) the $\bar{B}$ part is taken care of. The error contribution from the B part can be calculated from Eq (A.6), with $\epsilon \rightarrow \rho$:

$$\chi^{b}(q) = \left(\frac{\partial\rho(q)}{\partial\Sigma_{b}(\beta_{lo})} \cdot \sigma_{b}(\beta_{lo})\right)^2 + \left(\frac{\partial\rho(q)}{\partial\Sigma_{b}(\beta_{hi}-1)} \cdot \sigma_{b}(\beta_{hi}-1)\right)^2$$

$$\Updownarrow$$

$$\chi^{b}(q) = \left(\frac{\partial\rho(q)}{\partial\Sigma_{wr}(q)} \cdot \frac{\partial\Sigma_{wr}(q)}{\partial\Sigma_{b}(\beta_{lo})} \cdot \sigma_{b}(\beta_{lo})\right)^2 +$$

$$\left(\frac{\partial\rho(q)}{\partial\Sigma_{ri}(q)} \cdot \frac{\partial\Sigma_{ri}(q)}{\partial\Sigma_{b}(\beta_{hi}-1)} \cdot \sigma_{b}(\beta_{hi}-1)\right)^2 \qquad \text{(A.13)}$$

Using equations (5.5), (5.4) and (5.6) in Eq (A.13):

$$\chi^{b}(q) = \left(\frac{-\Sigma_{ri}(q)}{N^2} \cdot 1 \cdot \sigma_{b}(\beta_{lo})\right)^2 + \left(\frac{\Sigma_{wr}(q)}{N^2} \cdot (-1) \cdot \sigma_{b}(\beta_{hi}-1)\right)^2$$

$$\Updownarrow$$

$$\chi^b(q) = \frac{1}{N^4} \cdot \left([\Sigma_{ri}(q) \cdot \sigma_b(\beta_{lo})]^2 + [\Sigma_{wr}(q) \cdot \sigma_b(\beta_{hi} - 1)]^2 \right) \qquad \text{(A.14)}$$

Where $N = \Sigma_{ri}(q) + \Sigma_{wr}(q)$ is the number of classified patterns. The last step is to modify Eq (A.7) by substituting ρ for ϵ and then inserting the information from Eqs (A.12) and (A.14):

$$\sigma_\rho(q) = \sqrt{\chi^b(q) + \chi^{bb}(q)}$$

$$\Updownarrow$$

$$\sigma_\rho = \frac{\sqrt{{\Sigma_{wr}}^2 \cdot [\sigma_b(\beta_2)^2 + \sigma_{bb}(\beta_1)^2] + {\Sigma_{ri}}^2 \cdot [\sigma_b(\beta_1)^2 + \sigma_{bb}(\beta_2)^2]}}{N^2} \qquad \text{(A.15)}$$

Where q was omitted, for compactness reasons, $\beta_1 \leftrightarrow \beta_{lo}$ and $\beta_2 \leftrightarrow \beta_{hi} - 1$.

A.4 Outlining $\sqrt{\epsilon}\cdot(2\rho-1)$

In the latest DELPHI paper on B_s^0 mixing [20] the authors refer to the quantity $\sqrt{\epsilon}\cdot(2\rho-1)$ as "proportional to the statistical significance of a signal from oscillations".

To arrive at this quantity one can start by looking at the signal, which is limited by statistics in two ways: By how large a fraction of the events that have been classified, i.e. the efficiency, and by how many of the events that have been correctly classified, i.e. the purity.

Let ϵ be the efficiency, ρ the purity and $\bar{\rho}$ the impurity. The impurity comes from events wrongly tagged as signal events, thus $\bar{\rho} = 1-\rho$. To find the "true" signal from oscillations those events wrongly tagged as signal events must be subtracted from the tagged events:

$$s = \epsilon\cdot\rho - \epsilon\cdot\bar{\rho} = \epsilon\cdot(\rho-\bar{\rho}) = \epsilon\cdot(2\rho-1) \tag{A.16}$$

To get the statistical significance of a signal from oscillation, which is the aim of this section, the background must be included. The significance is usually defined as *signal divided by the square root of the background*, $A = s/\sqrt{b}$, but in a B-mixing analysis the signal will also be part of the background. Instead one can use the conservative modification that $A = s/\sqrt{s+b}$.

For a given analysis the signal and the background must add up to the efficiency, and in combination with Eq (A.16) one gets:

$$s+b = \epsilon \Rightarrow b = \epsilon - s = \epsilon\cdot(1-2\rho+1) = 2\epsilon\cdot(1-\rho) \tag{A.17}$$

Using the expressions for s and b, from Eqs (A.16) and (A.17):

$$A = \frac{s}{\sqrt{s+b}} = \frac{\epsilon\cdot(2\rho-1)}{\sqrt{\epsilon\cdot(2\rho-1)+2\epsilon\cdot(1-\rho)}} = \frac{\epsilon\cdot(2\rho-1)}{\sqrt{\epsilon}} = \sqrt{\epsilon}\cdot(2\rho-1)$$

This is the same quantity that DELPHI lists in their paper.

Appendix B

JetNet 3.0

Jetnet 3.0 [22] is a versatile Artificial Neural Network Package written in Fortran 77. With a total of 72 switches available one have much freedom in composing a neural network and training it.

B.1 Specific choices

Because the output from the net should be either negative or positive an activation function with that feature was needed: *tanh* was chosen.

According to the JetNet manual most of the problems in High Energy Physics seem to have very simple discrimination surfaces so that one hidden layer should be enough to encode a solution. Thus the "one hidden layer" structure was used throughout the analysis.

In section 4.5.1 the JetNet learning parameters, α and η, were mentioned. They are important in the process of finding a minimum value of χ^2, and the following update rules were adopted from an earlier DELPHI analysis[24]:

$$\eta_t = \eta_{t-1} \times \left(\frac{\eta_{min}}{\eta_{t-1}} \right)^{\kappa_\eta} \tag{B.1}$$

$$\alpha_t = \alpha_{t-1} \times \left(\frac{\alpha_{max}}{\alpha_{t-1}} \right)^{\kappa_\alpha} \tag{B.2}$$

Range of the parameters: $\eta \in [0.0001,0.05]$ and $\alpha \in [0.4,0.9]$. The constants have values κ_η=0.05 and κ_α=0.14 taken from the DELPHI analysis as well.

α and η are updated each epoch, which per JetNet default is after each 1000 patterns. If the training error is smaller than in the previous epoch the parameters are updated according to Eqs (B.1) and (B.2). But if the error increased η was increased by 20% (but was required to be below 0.01), and α lowered by 20% (but was required to be above 0.04).

B.2 CPU and RAM demanding training

Training an ANN is not a small task, it is usually both very CPU and RAM demanding. Typical nets in this analysis required between 20 Mb and 50 Mb RAM, mainly due to the large input array. The time spent on training the nets were usually of the order 10,000 to 20,000 CPU-seconds on a 91 Mb RAM, 266 MHz AlphaStation. But on smaller machines the training sessions more often had to compete with other user processes, thus swapping occured more frequently slowing down the training. In addition a lower clock frequency will easily double or triple the training time.

Though the above figures will vary from machine to machine, depending on the accessible RAM, a general statement like "training an ANN based on the `JetNet 3.0` package is very time consuming" should be underlined. A wrong statement in a piece of code and a days work may be useless.

Appendix C

Program listing

This appendix describes the main routines of the analysis and in a few cases give excerpts of important code. However, the standard JetNet 3.0 code [22], the Skelana structure [27] and Patchy technology [31] can be found elsewhere and will be omitted here.

C.1 General source code

To extract information from the DST's (identified by a `PDLINPUT` file) Patchy technology, with its CAR and CRA files, was used in a combination with Skelana. This was implemented in `extract.car` which used the quality cuts from section 5.1 during the extraction, and afterwards it stored the results as column-wise Ntuples in two hbook files: `netch.hbo` (training sample) and `tsample.hbo` (test sample).

The training was performed by `train.car` which reads the training data from `netch.hbo`. After the training session the final weight matrix of the neural net was dumped to a file named `jndat.dump`.

The testing was performed by `ntest.car` which reads the test data from `tsample.hbo` and the weights from `jndat.dump`. Test output were written directly to the screen so it had to be piped into a suitable file (`output.log`) during execution: `UNIX-prompt> ntest> output.log`

The jetcharge calculations were done with the help of a short CAR-file named `jetch.car` in which the hadronic selections were applied.

Each time changes were made to the net-structure or input variables the `train.car` and `ntest.car` files had to be modified. Thus after almost a year of work on the thesis several dozen versions of these files existed, only the latest versions can be found at: `http://www.fys.uio.no/~bor/physrc/`

C.2 NTCOPY

Both `ntest.car` and `train.car` have a common Fortran subroutine called `NTCOPY` which is invoked in the NTCOPY deck of the two CAR-files. This subroutine copies the desired information from the Ntuples, forms the input variables based on this information and put the results into suitable arrays for the JetNet training or testing.

In this listing `BCDE` is a common block defined in `extract.car`, where the entire Ntuple is declared and the hadronic selections are initialized according to the values in section 5.2. `pcut` is the cut value of the momentum, `crazyp` the value that momentum above beam energy is rescaled to and `trkcut` is the minimum number of charged tracks per hemisphere. The `datain` and `dataout` arrays are used to store the input variables and to feed the JetNet input and output arrays.

```
      SUBROUTINE NTCOPY(pos)
+CDE, BCDE.
      INTEGER pos, qq
      INTEGER nsoft, itrk
      REAL pqsum, ptsum, qbtag
*
*     Make cut on number of charged tracks and make sure
*     that at least one of the tracks in the hemisphere
*     has momentum above pcut
      IF (ntrack1.GE.trkcut.AND.p1(1).GT.pcut) THEN
*
*        Count the entry as accepted and normalize output value
         pos             = pos + 1
         dataout(1, pos) = qqbar1 / 5
*
*        Reset input values
         CALL vzero(datain(1, 1, pos), nvar*alltr)
*
*     Input values from hard tracks:
*     ------------------------------
*
         DO itrk = 1, alltr-1
            qq = q1(itrk)
*           Use only tracks with momentum above cut value
            IF (p1(itrk).GT.pcut) THEN
               IF (p1(itrk).GT.beamen) THEN
                  datain(1, itrk, pos) = qq*crazyp/beamen
               ELSE
                  datain(1, itrk, pos) = qq*p1(itrk)/beamen
               ENDIF
*           P_t information
               datain(2, itrk, pos)    = qq*pt1(itrk)/beamen
*           Btag information
               IF (btg1(itrk).LE.1.AND.btg1(itrk).GE.minb) THEN
                  datain(3, itrk, pos) = qq*log(btg1(itrk))/log(minb)
               ELSEIF (btg1(itrk).LT.minb) THEN
                  datain(3, itrk, pos) = qq
               ELSE
                  datain(3, itrk, pos) = 0.
                  print *,'===> btag above unity!! Value=', btg1(itrk)
               ENDIF
*
            ELSE
               datain(1, itrk, pos) = 0.
               datain(2, itrk, pos) = 0.
               datain(3, itrk, pos) = 0.
            ENDIF
         ENDDO
*
*     Input values from soft tracks:
*     ------------------------------
*
*     Use jetcharge to sum all soft tracks. 'soft tracks'
*     are defined as the remaining tracks when the (alltr-1)
*     number of hardest tracks have been used
```

```
         nsoft = ntrack1 - (alltr - 1)
         IF (nsoft.GT.0) THEN
            pqsum = 0.
            ptsum = 0.
            qbtag = 0.
            DO itrk = alltr, ntrack1
               qq = q1(itrk)
*              Sum up tracks with momentum above pcut
               IF (p1(itrk).GT.pcut) THEN
                  pqsum  = pqsum + (qq*p1(itrk)/beamen)
                  ptsum  = ptsum + (qq*pt1(itrk)/beamen)
                  IF (btg1(itrk).LE.1.AND.btg1(itrk).GE.minb) THEN
                     qbtag = qbtag + (qq*log(btg1(itrk))/log(minb))
                  ELSEIF (btg1(itrk).LT.minb) THEN
                     qbtag = qbtag + qq
                  ELSE
                     print *,'==> btag above unity! Value=', btg1(itrk)
                  ENDIF
               ENDIF
            ENDDO
*
*     Store the information for the soft track
            datain(1, alltr, pos) = pqsum
            datain(2, alltr, pos) = ptsum
            datain(3, alltr, pos) = qbtag
         ELSE
            datain(1, alltr, pos) = 0.
            datain(2, alltr, pos) = 0.
            datain(3, alltr, pos) = 0.
         ENDIF
*
      ENDIF
*
      RETURN
      END
```

C.3 abscut.f

Fortran subroutine written to be called from PAW. It takes two histograms as input, "id1" with the $\bar{B}$ distribution and "id2" with B. From those it computes the purity and efficiency which it stores in histograms 88 and 89. It also computes A_{max} and print some statistics when finished.

```
      SUBROUTINE abscut4(id1,id2)
      IMPLICIT NONE

************************************************************************
************************************************************************
*     Input:   id1 and id2 are two filled histograms with              *
*              the same number of channels (or bins) and               *
*              with range from -1.0 to +1.0                            *
*              id1 must contain the Bbar (OUT.LT.0) and id2            *
*              the B (OUT.GT.0) information                            *
*                                                                      *
*     Output:  Histograms id3 and id4, displaying the integrated       *
*              B and Bbar distributions. Histograms id5 and id6,       *
*              displaying the tagging purity and mean tagging          *
*              efficiencyiy as functions of the cuts on the            *
*              "netcharge". Thus pur(Q) is the purity obtained when    *
*              cutting on netcharge at -Q and +Q removing the          *
*              events inbetween.                                       *
*                                                                      *
*     Notes:   - The Delphi jetcharge is shifted, so this method       *
*              incorporates a Qshift. But currently the shift is       *
*              only 0.015, or of the order 1 bin!                      *
*                                                                      *
*                                                                      *
*     Created:  March 7 1996                                           *
*     Modified: June 17 1996                                           *
************************************************************************
************************************************************************
*
*
*
```

```
*********************************************************************
*                 Variable declaration                              *
*********************************************************************
*
      INTEGER id1, id2, id3, id4, id5, id6
      PARAMETER (id3=86, id4=87, id5=88, id6=89)
*
*     External functions
      LOGICAL hexist
*
*     Local variables
      INTEGER max_nx
      PARAMETER (max_nx=1000)
*
*     DELPHI has shifted the jetcharge |Q_mean_jet - 0.015|=0
      REAL Qshift
      PARAMETER (Qshift=0.02)
*
      CHARACTER*100 chtit1, chtit2
      INTEGER i, j, nmid, nzero, nhigh, nhem
      INTEGER nx1, ny1, nwt1, loc1,
     $        nx2, ny2, nwt2, loc2
      REAL nbp, nbbp, prt1, prt2, prt3, prt4
      REAL xmi1, xma1, ymi1, yma1,
     $     xmi2, xma2, ymi2, yma2
      REAL bdis(max_nx), bbdis(max_nx)
      REAL bbsum(max_nx), bsum(max_nx),
     $     errb(max_nx), erbb(max_nx)
      REAL tpur(max_nx), stp(max_nx)
      REAL teff(max_nx), ste(max_nx)
*
      INTEGER idmax
      REAL maxmix, tmix, errm, nchcut
*
      LOGICAL id1_exists, id2_exists
*
*
*
*********************************************************************
*         Read input histograms - check for errors                  *
*********************************************************************
*
*     Check that histograms exist in memory
      id1_exists = HEXIST(id1)
      IF (.NOT.id1_exists) THEN
         PRINT *, id1, ' does not exist'
         RETURN
      ENDIF
*
      id2_exists = HEXIST(id2)
      IF (.NOT.id2_exists) THEN
         PRINT *, id2, ' does not exist'
         RETURN
      ENDIF
*
*     Get histogram definition for id1
      CALL HGIVE(id1, chtit1, nx1, xmi1, xma1, ny1, ymi1,
     $           yma1, nwt1, loc1)

      IF (nwt1.GT.100) THEN
         PRINT *, 'Too many characters in title', nwt1
         RETURN
      ENDIF
*
      IF (nx1.GT.max_nx) THEN
         PRINT *, 'Too many bins: ', nx1
         RETURN
      ENDIF
*     Get histogram definition for id2
      CALL HGIVE(id2, chtit2, nx2, xmi2, xma2, ny2, ymi2,
     $           yma2, nwt2, loc2)

      IF (nwt2.GT.100) THEN
         PRINT *, 'Too many characters in title', nwt2
         RETURN
      ENDIF

      IF (nx2.GT.max_nx) THEN
         PRINT *, 'Too many bins: ', nx2
         RETURN
      ENDIF
*
      IF (nx2.NE.nx1) THEN
```

```
         PRINT *, 'Error: Input histos have different ranges!'
         RETURN
      ENDIF
*
*     Get contents of input histograms
      CALL HUNPAK(id1, bbdis, ' ', 0)
      CALL HUNPAK(id2, bdis, ' ', 0)
*
*
*
*
**********************************************************************
*   Compute the desired quantities:                                  *
*                                                                    *
*   Purity           = (Nr of correct tagged) / (Nr of tagged)       *
*   Efficiency       = (Nr of tagged) / (all tagged + untagged)      *
*   Mixing Amplitude = SQRT(Efficiency) * (2*purity - 1)             *
**********************************************************************
*
*     Find the middle bin of the nx1 channels of the input histos
      IF (MOD(nx1,2) .EQ. 1) THEN
         nmid = (nx1-1)/2
         PRINT *,'*** MODULUS(nx1,2).EQ.1 ***'
      ELSE
         nmid = nx1/2
      ENDIF
*
*     The shifted zero bin, |Q_mean_jet - shift| = 0, can be
*     calculated by adding the shift to the middle bin (nmid)
      nzero = nmid + INT(nmid * Qshift)
      nhigh = 1. + Qshift
*
*
*
*     Accumulate b- and bbar-histogram contents i.e. sum
*     up the b and bbar distributions
      bbsum(1) = bbdis(1)
      bsum(1)  = bdis(1)
*
      DO i = 1, nx1-1
         bbsum(i+1) = bbsum(i) + bbdis(i+1)
         bsum(i+1)  = bsum(i)  + bdis(i+1)
      ENDDO
*
      IF (bbsum(nx1).LE.0.OR.bsum(nx1).LE.0) THEN
         PRINT *, 'No events! '
         RETURN
      ENDIF
*
*     The total number of hemispheres in the histograms
      nhem = bsum(nx1) + bbsum(nx1)
*
* --------------------
*
*     Compute the tagging purity, aka the mean "correct
*     tagging efficiency"
      j = 0
      DO i = nzero, nx1-1
         j = j + 1
*     Correct tagged (j = i - nzero + 1)
         nbp  = bbsum(nzero-j) + (bsum(nx1) - bsum(i))
*     All tagged (i.e. outside the cut)
         nbbp = nbp + bsum(nzero-j) + (bbsum(nx1) - bbsum(i))
*
         IF (nbbp.GT.0) THEN
            tpur(j) = nbp / nbbp
         ELSE
            tpur(j) = 0.
         ENDIF
      ENDDO
*
*     Now only events on the negative side are left since
*     the shifted mean is on the positive side and we have
*     summed symmetrically around the "shifted mean bin"
      j = j + 1
      DO i = j, nzero-1
*     Correct tagged
         nbp  = bbsum(nzero-i)
*     All tagged
         nbbp = nbp + bsum(nzero-i)
*
         IF (nbbp.GT.0) THEN
            tpur(i) = nbp / nbbp
```

```
         ELSE
            tpur(i) = 0.
         ENDIF
      ENDDO
*
* --------------------
*
*     Compute the "mean tagging efficiency" when nzero is
*     the shifted zero bin of the mean distribution
      i = 0
      DO j = nzero, 2, -1
         i = i + 1
*        Sum correct tagged Bbar and wrong tagged B below -Q_cut
         prt1 = bsum(nzero-i) + bbsum(nzero-i)
         IF ((nzero+i).LE.nx1) THEN
*           There are still some bins left above +Q_cut which will
*           give the number of correct tagged B's and wrong Bbar
            prt2 = bsum(nx1)  - bsum(nzero+i-1) +
     $             bbsum(nx1) - bbsum(nzero+i-1)
         ELSE
            prt2 = 0.
         ENDIF
         teff(i) = (prt1 + prt2) / nhem
      ENDDO
      teff(nzero) = 0.
*
* --------------------
*
*     Find the maximum value of the statistical significance
*     of a signal from oscillation: SQRT(eff) * (2*pur - 1)
      maxmix = 0.
      DO i = 1, nzero
         tmix = SQRT(teff(i)) * (2*tpur(i) - 1)
         IF (maxmix.LT.tmix) THEN
            idmax  = i
            maxmix = tmix
         ENDIF
      ENDDO
      nchcut = real(idmax)/real(nmid)
*
*
*
*
**************************************************************************
*               Compute the errors                                       *
**************************************************************************
*
*
*     The errors in bbsum and bsum (the integrated distributions) is
*     given by Binomial statistics with variance n(N-n)/N where N
*     is the total number and n the number of events integrated to
*     the bin where the error is calculated.
*     errb and erbb is the errors in the integrated b and bbar
*     distributions respectively, both for bin i.
*
      DO i = 1, nx1
         erbb(i) = SQRT( bbsum(i) * (bbsum(nx1) - bbsum(i)) /
     $                   bbsum(nx1) )
         errb(i) = SQRT( bsum(i)  * (bsum(nx1)  - bsum(i))  /
     $                   bsum(nx1)  )
      ENDDO
*
* ------------------
*
*     The errors in the mean tagging purity
      j = 0
      DO i = nzero, nx1-1
         j = j + 1
*     Correct tagged patterns
         nbp  = bbsum(nzero-j) + (bsum(nx1) - bsum(i))
*     Wrong tagged patterns
         nbbp = bsum(nzero-j)  + (bbsum(nx1) - bbsum(i))
*
         IF ((nbbp+nbp).GT.0) THEN
            prt1   = nbbp**2
            prt2   = (errb(i)**2) + (erbb(nzero-j)**2)
            prt3   = nbp**2
            prt4   = (errb(nzero-j)**2) + (erbb(i)**2)
            stp(j) = SQRT( (prt1*prt2) + (prt3*prt4) ) / (nbbp+nbp)**2
            IF (stp(j).GT..75) THEN
*              No reason to have error bars greater than 1.5
               stp(j) = 0.75
            ENDIF
```

```
         ELSE
            stp(j) = 0.
         ENDIF
      ENDDO
*
      j = j + 1
      DO i = j, nzero-1
*     Correct tagged
         nbp  = bbsum(nzero-i)
*     Wrong tagged
         nbbp = bsum(nzero-i)
*
         IF ((nbbp+nbp).GT.0) THEN
            prt1   = nbbp**2
            prt2   = (erbb(nzero-i))**2
            prt3   = nbp**2
            prt4   = (errb(nzero-i))**2
            stp(i) = SQRT( (prt1*prt2) + (prt3*prt4) ) / (nbbp+nbp)**2
         ELSE
            stp(i) = 0.
         ENDIF
      ENDDO
*
* ------------------
*
*     The errors in the mean tagging efficiency
      i = 0
      DO j = nzero, 2, -1
         i = i + 1
         prt1 = (erbb(nzero-i)**2) + (errb(nzero-i)**2)
         IF ((nzero+i).LE.nx1) THEN
            prt2 = (errb(nzero+i-1)**2) + (erbb(nzero+i-1)**2)
         ELSE
            prt2 = 0.
         ENDIF
         ste(i) = SQRT( prt1 + prt2 ) / nhem
      ENDDO
*
* ------------------
*
*     Normalize the integrated distributions (the error
*     first since bsum(nx1) and bbsum(nx1) will be 1 after
*     normalization of the B and Bbar distributions!)
      DO j = 1, nx1
         erbb(j)  = erbb(j)  / bbsum(nx1)
         bbsum(j) = bbsum(j) / bbsum(nx1)
*
         errb(j) = errb(j) / bsum(nx1)
         bsum(j) = bsum(j) / bsum(nx1)
      ENDDO
*
* ------------------
*
*     The errors in the calculated mixing amplitude:
      prt1 = ((2*tpur(idmax)) - 1) ** 2
      prt2 = ste(idmax) ** 2
      prt3 = (2*teff(idmax)) ** 2
      prt4 = stp(idmax) ** 2
      errm = SQRT( prt1*prt2 + prt3*prt4 )
*
*
*
**********************************************************************
*       Prepare output and book the histograms                       *
**********************************************************************
*
*
*     Book histograms
      CALL hbook1(id3,'Integrated Bbar distribution',nx1,-1.,1.,0.)
      CALL hbook1(id4,'Integrated B distribution',nx1,-1.,1.,0.)
      CALL hbook1(id5,'Purity as function of cut on netcharge',
     $               nzero,0.,nhigh,0.)
      CALL hbook1(id6,'Mean tagging efficiency vs cut on netcharge',
     $               nzero,0.,nhigh,0.)
*
*     Fill histograms together with the calculated errors
      CALL HPAK(id3,  bbsum)
      CALL HPAKE(id3, erbb)
*
      CALL HPAK(id4,  bsum)
      CALL HPAKE(id4, errb)
*
      CALL HPAK(id5,  tpur)
```

```
      CALL HPAKE(id5, stp)
*
      CALL HPAK(id6,  teff)
      CALL HPAKE(id6, ste)
*
*     List the mixing information and the histogram information
      WRITE(*,110) ' '
      WRITE(*,110) '------------- routine: abscut4.f ------------'
      WRITE(*,111) 'Max mixing amplitude',maxmix,'with error',errm
      WRITE(*,112) 'Occurs for a cut',nchcut,'on absolute Netcharge'
      WRITE(*,111) 'Purity=',tpur(1),'at eff=100, err=',stp(1)
      WRITE(*,110) ' '
      WRITE(*,110) '===> Output histograms: 86,87,88 and 89'
      WRITE(*,110) '--------- rostad; modified 17.06.96 ---------'
      WRITE(*,110) ' '
*
 110  FORMAT('',TR10,A)
 111  FORMAT('',TR10,A,TR1,F6.4,TR1,A,TR1,F6.4)
 112  FORMAT('',TR10,A,TR1,F5.3,TR1,A)
*
      RETURN
      END
```

Appendix D

Glossary

2VTX: Is short for Secondary Vertex, the point where a long lived particle from PV, like a B meson, decays and leaves a debris of new particles.

CUT: A short and handy word used throughout the HEP community. To pose a "cut" means to select data restricted by certain criterias.

DST: A Data Summary Tape is the format in which simulated and real data, from DELPHI, are stored.

HADRON: Particles containing quarks are commonly referred to as hadrons. Thus both mesons (with 2 quarks) and baryons (with 3 quarks) are hadrons.

HEP: High Energy Physics. Today Elementary Particle Physics and HEP is essentially the same, because most of the particles labeled as elementary only can be created with the help of huge accelerators, i.e. at high energies, because of their large mass.

KAON: Is the common name for mesons containing one strange quark.

MESON: Means a light particle. This reflects that all particles labeled mesons contain only two quarks while baryons, with three quarks, are usually heavier.

NETCHARGE: Was invented for this thesis; "netcharge" is the short for *neural NETwork jetCHARGE-like output*, which should hint towards the similarities between this method and Jetcharge.

NTUPLE: Is a common way of storing large amounts of data in physics analysises in the HEP community. It can be viewed as a matrix where, for example, each row represents an event and each column a variable (charge, energy, momentum, etc). This is a row-wise Ntuple. In a column-wise Ntuple the elements of each column are stored sequentially, thus a column can be viewed as one event.

QUARK: Is the building blocks of atomic matter and of all non-elementary particles (hadrons). The name is said to be derived from the line "Three quarks for Muster Mark!" from James Joyce's *Finnegans Wake* (1939), but Murray Gell-Mann have said that he discovered this connection only after the name was thought up. An alternative explanation for the name "quark": At the time, when Gell-Mann put forward his theory, most physicists regarded the quarks as interesting mathematical objects but questioned their physical reality. Hence **qu**estion m**ark**!

PAW: Physics Analysis Workstation is an important software tool in the HEP community, it helps setting up data and visualize the results in suitable plots or histograms. Huge Ntuples with data are treated with the greatest ease by PAW.

PV: Is short for Primary Vertex, a.k.a. the interaction point, which is the point where particles from the two beams (e^+ and e^-, for example) collide, giving rise to an interesting event.

SKELANA: Is the SKELeton ANAlysis program used in DELPHI. With this program much of the basic work, like reading DST files and filling the standard common blocks with data, is done by the program freeing the user to concentrate on cut selections and physics problems.

Index

Bibliography

[1] Frank J. Blatt "Modern Physics" **Eq. (6.3)**, ©1992 McGraw-Hill, Inc. **ISBN 0-07-112918-9**

[2] Donald H. Perkins "Introduction to High Energy Physics (3.edition)", **Eq (2.2)**. ©1987 Addison-Wesley Publishing Company, Inc. **ISBN 0-201-12105-0**

[3] The DELPHI Collaboration "Search for New Phenomena Using Single Photon Events in the DELPHI Detector at LEP", 10 January 1996, **CERN-PPE/96-03**

[4] Peter B. Renton "Review of Experimental Results on Precision Tests of Electroweak Theories", May 13 1996, **CERN-PPE/96-63**

[5] Bob Jacobsen "Top Mass From Electroweak Measurements", June 27 1994, **CERN-PPE/94-97**

[6] The DELPHI Collaboration
"The DELPHI Detector at LEP", 9 November 1990. *Nuclear Instruments and Methods in Physics Research* **A303** (1991) **p.233-276**
"Performance of the DELPHI Detector", 30 June 1995, **DELPHI 95-112 PHYS 547**

[7] Donald H. Perkins "Introduction to High Energy Physics (3.edition)", **Chapter 5**. ©1987 Addison-Wesley Publishing Company, Inc. **ISBN 0-201-12105-0**

[8] F.Halzen and A.D.Martin "Quarks & Leptons", **Chapter 10**. ©1984 John Wiley & Sons, Inc. **ISBN 0-471-81184-4**

[9] Particle Data Group *Physical Review D - Particles and fields*, 1 August 1994, part 1, Review of Particle Properties. **p.1319-22**

[10] Gell-Mann and Pais *Physical Review* **97**, 1387 (1955)

[11] Particle Data Group *Physical Review D - Particles and fields*, 1 August 1994, part 1, Review of Particle Properties. **p.1632-3** and **p.1640-1**

[12] Paolo J. Franzini "$B\bar{B}$ mixing: a review of recent progress". Published in Physics Report **173** (1989) **p.11-13**

[13] Frank Close, Michael Marten and Christine Sutton "The Particle Explosion", **p.186-8**. Published by Oxford University Press 1987. **ISBN 0 19 853999 1 (pbk)**

[14] Particle Data Group *Physical Review D - Particles and fields*, 1 August 1994, part 1, Review of Particle Properties. **p.1195**

[15] Particle Data Group *Physical Review D - Particles and fields*, 1 August 1994, part 1, Review of Particle Properties. **p.1207-1210**

[16] Vernon Barger and Roger Phillips "Collider Physics", **chapter 3.7.3**, ©1987 Addison-Wesley, **ISBN 0-201-05876-6**

[17] Guennadi V. Borisov "Lifetime Tag of events $Z^0 \rightarrow b\bar{b}$ with the DELPHI detector. AABTAG program." **DELPHI 94-125 PROG 208**. Geneva 1994.

[18] H.Albrecht et al., *Phys.Letters* **B192** (1987) **p.245**

[19] Sheldon Stone et al "B DECAYS - revised 2nd edition", **p.449-469**, ©1994 World Scientific Publishing Co. **ISBN 981-02-1836-2**

[20] M.Canepa, O.Kouznetsov, F.Parodi, P.Paganini, P.Roudeau, A.Stocchi, A.Ouraou and I.Ripp "Search for $B_s^0 - \overline{B_s^0}$ oscillations", 3 April 1996, **DELPHI 96-45 PHYS 617**.

[21] The ALEPH Collaboration "Limit on B_s^0 Oscillation Using a Jet Charge Method", 7 June 1995, **CERN-PPE/95-84**

[22] Carsten Peterson, Thorsteinn Rögnvaldsson and Leif Lönnblad "JETNET 3.0 - A Versatile Artificial Neural Network Package", December 1993, **CERN-TH.7135/94**

[23] C.Bortolotto, A.De Angelis, N.De Groot and J.Seixas "Neural Networks in Experimental High-Energy Physics". Submitted to *International Journal of Modern Physics C* Vol.3, No.4 (1992) **p.733-771**

[24] The DELPHI Collaboration "Classification of the hadronic decays of the Z^0 into b and c quark pairs using a neural network". Submitted to *Physics Letters B* 295 (1992) **p.383-395**

[25] The DELPHI Collaboration "Measurement of the Mean Charged Multiplicity in $Z^0 \rightarrow b\bar{b}$ Events", June 1 1994 **DELPHI 94-64 PHYS 385**

[26] The DELPHI Collaboration
"DELSIM user's guide", 10 July 1989, **89-67 PROG 142**.
"DELSIM Reference Manual", 1 September 1989, **89-68 PROG 143**

[27] Tzanko Spassoff and Nikolai Smirnov "SKELANA - skeleton analysis program". DELPHI 1995.

[28] Torbjörn Sjöstrand "PYTHIA 5.7 and JETSET 7.4 - Physics and Manual", **CERN-TH.7112/93**, December 1993 (revised August 1994), **p.280-1**

[29] Richard Larsen and Morris Marx *An introduction to Mathematical Statistics and Its Applications*, 2nd edition, **Eq (3.8.4)**, **Ex 3.10.3** and **p.238-9** ©1986 Prentice-Hall **ISBN 0-13-487174-X**

[30] Wilhelm Løchstøer *Fysiske Målinger - Elementær måleteori og usikkerhetsregning*, 1983 Fysisk Institutt, Universitetet i Oslo, **p.20.b**

[31] H.J. Klein and J. Zoll "PATCHY. Reference Manual - revised for version 4.13", CERN Program Library. Marc 1988.