Constraints to an Extra Vector Gauge Boson by Anaysis of the Process $e^+e^- \to \gamma\nu\bar{\nu}$

Dietrich Rothe

Advisor: I. Schienbein

June 11 2007

Contents

Chapter 1

Introduction

1.1 Location of Internship

I am doing my internship at the Laboratoire de Physique Subatomique et de Cosmologie (LPSC) at Grenoble. This institute focusses on experimental and theoretical research in particle physics, astrophysics, nuclear and hadronic matter and developement of particle detectors. It has in its employ about 200 researchers, engineers and technical experts and recieves about 60 student trainees each year. The LPSC also collaborates with CERN and Fermilab. In my workgroup, research subjects are thereotical and phenomenological particle physics, including Super Symmetry considerations.

1.2 My Subject

My subject is to analyze data of the process $e^+e^- \rightarrow \gamma, <$ invisible $>$ which has been recoreded at the Large Electron-Positron Collider (LEP) at CERN. By $<$ invisible $>$ I mean that I will focus on data where besides a photon there is produced something else which is not detected, what we know because of missing energy and momentum. Normally this is the Z boson of the standard model, which rapidly decays into an antineutrino-neutrino pair of the same generation, and these cannot be detected. The Z boson can also decay into visible particles, but those processes are not considered. However if we assume there is another neutral gauge boson (from beyond the standard model) which we will call Z', there might also occur the process $e^+e^- \rightarrow \gamma, Z'$ followed by $Z' \rightarrow \nu, \bar{\nu}$. The goal is to find slight deviations from SM

predictions in the data which correspond to this process. If we assume the additional reaction takes place, we can use the data to find contraints on the parameters of the Z' (these parameters are mass and axial and vector coupling constants to other particles).

I will start with a short overview of the standard model and motivations for an additional vector gauge boson Z' in extended theories. Then I will analytically calculate a simplified cross section of the SM process $e^+e^- \rightarrow \gamma, Z$. This is done partly by hand and partly with the mathematica Tracer package. The next step is numerical integration of the cross section and graphical comparison which the data. The final step, which isn't yet achieved, will be the construction of exclusion plots for Z' parameters.

A considerable part of my work was learning the usage of computer tools like ROOT and Tracer and writing the evaluation and analyzation programs, which are partly included in the appendix.

1.3 The Standard Model

The standard model of electroweak interactions has gauge symmetry $SU(2)_L \times U(1)_Y$. At low energy, this symmetry breaks to $U(1)_{em}$ symmetry of electrodynamics. An often used model to illustrate symmetry breaking is a 3-dimensional space filled with a ferromagnetic material. At high energy, above Curie temperature T_C, the system becomes paramagnetic and is therefore invariant under $SO(3)$ transformations. Below T_C the magnetic dipoles align and symmetry breaks to $SO(2)$, which is a subgroup of $SO(3)$. In general one speaks of symmetry breaking when the ground state of a system has lower symmetry than the system's Lagrangien.

There is an extended SM which successfully describes the strong and electroweak interactions with a $SU(3)_c \times SU(2)_L \times U(1)_Y$ gauge group. Many physicits believe there exists a model describing all fundamental interactions (including gravitation) in such a way. Those theories are called Grand Unified Theories (GUT). A gauge group G_{GUT} must contain the SM and break to $SU(3)_c \times SU(2)_L \times U(1)_Y$ at $E \ll E_{GUT}$. GUT predict the proton must decay, similar to β decay of electroweak theory. Present experiments on proton decay (which has not yet been observed) give the condition $E_{GUT} > 10^{15}\ GeV$. This bound is still much smaller than the Planck mass $M_P = \sqrt{\hbar c/G} \approx 1.2\ 10^{19}\ GeV$, which describes the energy scale at which gravitation is expected to become as strong as the other interactions.

It was shown that the smallest gauge group G which can contain the SM, $SU(5)$, cannot be used to describe all interactions in a unified way. All larger groups containing the SM predict at least one neutral gauge boson (Z'). Its mass is constrained by energies of symmetry breaking: $E_{weak} < M'_Z < E_{GUT}$.

Extensive research has be done on this Z', in particular concerning stronger contraints to its parameters. This is also our goal.

Here I will try to give a brief description of the electroweak standard model so far as needed for my work.The standard model was introduced in the middle of the 1960s in works by S. Glashow, A. Salam and S. Weinberg aiming to describe electromagnetic and weak interactions in an unified theory. It is based on the gauge theory by C. N. Yang and R. L. Mills who in 1954 introduced the local gauge invariance of the weak isospin. In the SM electromagnetic and weak interaction appear as two different aspects of one single interaction, and are mediated by the γ respective $W^{\pm}$ and Z^0 bosons.

All elementary fermions (leptons and quarks) are sensible to weak interaction. In order to describe the interaction the concept of charge-raising, charge-lowering and unchared currents is introduced.

This is analogous to the describtion of e.g. electron-electron scattering mediated by a γ as in QED, where the moving electrons are regarded as currents. By treating charged leptons and neutrinos as two different aspects of the same particle, it is possible to regard the reaction of muon decay $\mu^+ \rightarrow e^+\nu_e\bar{\nu_\mu}$ as charge-lowering μ-type current interacting with a charge-raising electron-type current via a W^+ or W^- boson. The weak interaction violates parity, only left-handed matter particles and right-handed antimatter particles interact by charged currents. In fact, there are only left-handed neutrinos and right-handed antineutrinos. One of the first historical exper-

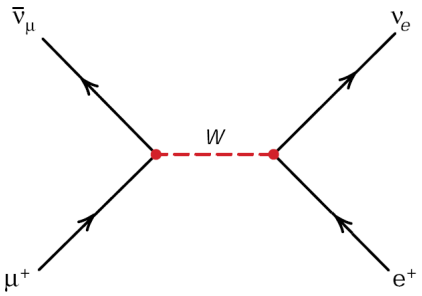

Figure 1.1: the muon decay $\mu^+ \rightarrow e^+\nu_e\bar{\nu_\mu}$ can be regarded as interaction of charge-changing currents.

iments that verified the parity violation studied β-transitions of polarized cobalt:

$$^{60}Co \rightarrow {}^{60}Ni^{*} + e^{-} + \bar{\nu}_e.$$

The spins of the cobalt nuclei were aligned by an external magnetic Field $\vec{B}$ and an asymmetry in the direction of the emitted electrons was observed. On reversal of $\vec{B}$, the asymmetry also switched. It is important that $\vec{B} \propto \vec{I} \times \vec{e}_r$ is an axial variable, that is, it does not change its sign if parity changes, because it is composed of two vector variables. We conclude there must be an axial coupling of weak interactions (in addition to a vector coupling constant).

In the γ-algebra of 4×4 Dirac γ matrices used in QED and its extensions, a vector variable like $\bar{\psi}\gamma\psi$ can be made axial by inserting a γ^5 like $\bar{\psi}\gamma\gamma^5\psi$ because of the anticommuting relation $\{\gamma^5, \gamma^\mu\} = 0$. The projector on left-handed particles can be written as $\frac{1}{2}\gamma^\mu(1-\gamma^5)$. These structures can be seen in the vertex factors of the feynman rules that I will use later. The vertex is surrounded by 4-dimensional spinors u and v used for ingoing fermions respective antifermions and their adjunct counterparts denoted by a bar ($\bar{u} = u^\dagger\gamma^0$) used for outgoing (anti)fermions.

Chapter 2

Calculations

2.1 Analytical Cross Section Calculation in Standard Model

The calculation of the matrix element $\mathcal{M}_{fi} =< f|\mathcal{M}|i >$ that describes the electroweak interaction between particles resulting in the final state f and having initial state i is carried out according to rules developed by R. P. Feynman in 1949. A Feynman diagram shows the incoming particles on the left side and outgoing particles on the right side. Particles meet in points called vertices, and vertices are connected by lines called propagators. The matrix element is built of vertex factors surrounded by spinors of their in- and outgoing particles and propagator factors. Antiparticles are formally treated as particles with negative impulse propagating backwards in time.

We will treat the Z' like it was an Z and for the first also use the same coupling parameters. The vertex factor associated with an electron-electron-Z coupling (like the lower black circle in fig. 2.1) is $-\frac{ig}{\cos\theta_W}\gamma^\mu\frac{1}{2}\left(c_V^f - c_A^f\gamma^5\right)$, where g is the weak charge, θ_W the Weinberg mixing angle (describing the Mixing of B and W^0 bosons to Z^0 and γ bosons), and c_V^f and c_A^f are vector and axial coupling constants which differ for left- and right-handed fermions. The vertex for electron-electron-γ coupling is $ie\gamma^\mu$ (for charge -1).

We set $V = -\frac{g}{2\cos\theta_W}c_V^f$ and $A = -\frac{g}{2\cos\theta_W}c_A^f$, so V and A are real and the vertex rule for weak neutral couplings becomes $i\gamma^\mu(V - A\gamma^5)$. By combining vertices surrounded by in- and outgoing spinors and the electron propagator $\frac{i}{\not{p}-m}$ we get

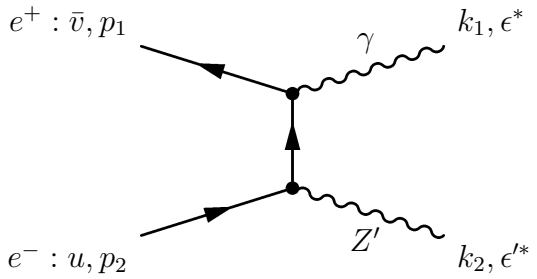

Figure 2.1: Feynman diagram for $e^+e^- \to \gamma Z'$ (t channel, $\mathcal{M}_1$)

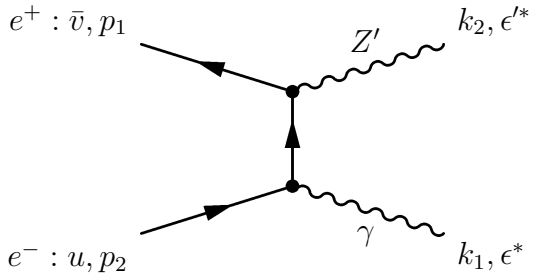

Figure 2.2: Feynman diagram for $e^+e^- \to \gamma Z'$ (u channel, $\mathcal{M}_2$)

$$-i\mathcal{M}_1 = \bar{v}(p_1)(ie\gamma^\mu)\epsilon^*_\mu \frac{i}{\not{p}_2 - \not{k}_2 - m}\epsilon'^*_\nu i\gamma^\nu(V - A\gamma^5)u(p_2)$$

Because we intend to study high energies we may neglect the electron mass, and using $t = (k_1 - p_1)^2 = (p_2 - k_2)^2$

$$\begin{aligned} -i\mathcal{M}_1 &= -\frac{ie}{t}\epsilon^*_\mu \epsilon'^*_\nu \bar{v}\gamma^\mu(\not{k}_1 - \not{p}_1)\gamma^\nu(V - A\gamma^5)u \\ i\mathcal{M}^*_1 &= \frac{ie}{t}\epsilon_{\mu'}\epsilon'_{\nu'}\bar{u}(V + A\gamma^5)\gamma^{\nu'}(\not{k}_1 - \not{p}_1)\gamma^{\mu'}v \end{aligned}$$

As the γ is a real photon, we can simplify with $\sum_{polariz.} \epsilon_{\mu'}\epsilon^*_\mu = -g_{\mu'\mu}$. For the massive Z' the relation is $\sum_{polariz.} \epsilon'_{\nu'}\epsilon'^*_\nu = -g_{\nu'\nu} + \frac{1}{M_Z^2}k_{2,\nu'}k_{2,\nu}$. The in-going particles can have any spin state. Spin averaging thus gives (note the 1/4 factor)

$$\begin{aligned} \overline{|\mathcal{M}_1|^2} &= \frac{-e^2}{4t^2}(-g_{\nu'\nu} + \frac{1}{M_Z^2}k_{2,\nu'}k_{2,\nu}) \cdot \\ &\quad \cdot \sum_{spins\ s,s'} \epsilon'_{\nu'}\epsilon'^*_\nu \bar{u}^{s'}(V + A\gamma^5)\gamma^{\nu'}(\not{k}_1 - \not{p}_1)\gamma_\mu v^s \bar{v}^s \gamma^\mu(\not{k}_1 - \not{p}_1)\gamma^\nu(V - A\gamma^5)u^{s'} \\ &= \frac{e^2}{4t^2}(g_{\nu'\nu} - \frac{1}{M_Z^2}k_{2,\nu'}k_{2,\nu})Tr\left[\not{p}_2(V + A\gamma^5)\gamma^{\nu'}(\not{k}_1 - \not{p}_1)(-2\not{p}_1)(\not{k}_1 - \not{p}_1)\gamma^\nu(V - A\gamma^5)\right] \end{aligned}$$

The other topology for $e^+e^- \to \gamma, Z$ is shown in fig. 2.2.

$$\begin{aligned} -i\mathcal{M}_2 &= \bar{v} i\gamma^\mu (V - A\gamma^5)\epsilon'^*_\mu \frac{i}{\not p_2 - \not k_1}\epsilon^*_\nu (ie\gamma^\nu)u \\ &= -\frac{ie}{u}\epsilon^*_\nu \epsilon'^*_\mu \bar{v}\gamma^\mu (V - A\gamma^5)(\not k_2 - \not p_1)\gamma^\nu u \\ i\mathcal{M}_2^* &= \frac{ie}{u}\epsilon_{\nu'}\epsilon'_{\mu'}\bar{u}\gamma^{\nu'}(\not k_2 - \not p_1)(V + A\gamma^5)\gamma^{\mu'}v \end{aligned}$$

$$\overline{|\mathcal{M}_2|^2} = \frac{e^2}{4u^2}(g_{\mu'\mu} - \frac{1}{M_Z^2}k_{2,\mu'}k_{2,\mu})Tr\left[\not p_1\gamma^\mu(V - A\gamma^5)(\not k_2 - \not p_1)\gamma^\nu \not p_2\gamma_\nu(\not k_2 - \not p_1)(V + A\gamma^5)\gamma^{\mu'}\right]$$

$$= \frac{e^2}{4u^2}(g_{\mu'\mu} - \frac{1}{M_Z^2}k_{2,\mu'}k_{2,\mu})Tr\left[\not p_1\gamma^\mu(V - A\gamma^5)(\not k_2 - \not p_1)(-2\not p_2)(\not k_2 - \not p_1)(V + A\gamma^5)\gamma^{\mu'}\right]$$

for this I exchanged $\nu' \leftrightarrow \mu'$ in $\mathcal{M}_2^*$:

$$\overline{\mathcal{M}_1\mathcal{M}_2^*} = \frac{e^2}{4tu}\epsilon'^*_\nu\epsilon'_{\nu'}\epsilon^*_\mu\epsilon_{\mu'}[\bar{v}\gamma^\mu(\not k_1 - \not p_1)\gamma^\nu(V - A\gamma^5)u][\bar{u}\gamma^{\mu'}(\not k_2 - \not p_1)(V + A\gamma^5)\gamma^{\nu'}v]$$

$$= \frac{e^2}{4tu}(g_{\nu'\nu} - \frac{1}{M_Z^2}k_{2,\nu'}k_{2,\nu})Tr\left[\not p_1\gamma^\mu(\not k_1 - \not p_1)\gamma^\nu(V - A\gamma^5)\not p_2\gamma_\mu(\not k_2 - \not p_1)(V + A\gamma^5)\gamma^{\nu'}\right]$$

Tracer is a Mathematica package which implements γ-algebra rules and can simplify above expressions. For the Tracer code (see appendix) we use the Mandelstam variables

$$\begin{aligned} s = \ & (p_1 + p_2)^2 = 2p_1p_2 \\ t = \ & (p_1 - k_1)^2 = -2p_1k_1 \\ u = \ & (p_1 - k_2)^2 = -2p_1k_2 + M_Z^2 \end{aligned}$$

The Tracer result is:

$$\begin{aligned} \overline{|\mathcal{M}|^2} &= \overline{|\mathcal{M}_1|^2} + 2Re[\overline{\mathcal{M}_1\mathcal{M}_2^*}] + \overline{|\mathcal{M}_2|^2} \\ &= \frac{8\pi\alpha(A^2 + V^2)\left(2M_Z^4 + t^2 + u^2 - 2M_Z^2(t + u)\right)}{tu} \\ &= \frac{8\pi\alpha(A^2 + V^2)\left(M_Z^4 + s^2 - 2tu\right)}{tu} \end{aligned} \tag{2.1}$$

Kinetic substitutions: In CMS with $s = (2E)^2$, we have

$$\begin{aligned} p_1 &= (E, \vec{p_i}) = \frac{\sqrt{s}}{2}(1, \vec{e}_z) \\ p_2 &= (E, -\vec{p_i}) = \frac{\sqrt{s}}{2}(1, -\vec{e}_z) \end{aligned}$$

if we neglect the electron mass. In the final state we have $k_1 = (|\vec{p}|, \vec{p})$ for the photon and $k_2 = (\sqrt{M_Z^2 + \vec{p}^2}, -\vec{p})$ for the Z', from what we get

$$s = \left(|\vec{p}| + \sqrt{M_Z^2 + \vec{p}^2} \right)^2 \Rightarrow (\sqrt{s} - |\vec{p}|)^2 = M_Z^2 + \vec{p}^2 \Rightarrow$$

$$|\vec{p}| = \frac{|s - M_Z^2|}{2\sqrt{s}} \tag{2.2}$$

$$t = (\frac{\sqrt{s}}{2} - |\vec{p}|, \frac{\sqrt{s}}{2}\vec{e}_x - \vec{p})^2 = \vec{p}^2 + \frac{s}{4} - |\vec{p}|\sqrt{s} - (\vec{p}^2 + \frac{s}{4} - \sqrt{s}p_x) = \sqrt{s}(-|\vec{p}| + p_x)$$

The deflection angle θ to the beam is $\cos\theta = \frac{p_x}{|\vec{p}|} \Rightarrow$

$$t = \frac{1}{2}(s - M_Z^2)(\cos\theta - 1) \tag{2.3}$$

From $s + t + u = M_Z^2$ we now get

$$u = \frac{1}{2}(s - M_Z^2)(-\cos\theta - 1) \tag{2.4}$$

The $\cos\theta$-dependant cross section is given by

$$\begin{aligned} \frac{d\sigma}{d\cos\theta}\Big|_{CM} &= \frac{1}{64\pi}\frac{p_f}{p_i s}|\mathcal{M}|^2 \\ &= \frac{1}{64\pi}\frac{|s - M_Z^2|}{s}|\mathcal{M}|^2 \end{aligned} \tag{2.5}$$

with the initial momentum $p_i = \sqrt{s}/2$ and the final momentum p_f from (2.2). The final result is

$$\frac{d\sigma}{d\cos\theta}\Big|_{CM} = \frac{\alpha(A^2 + V^2)}{2s^2(s - M_Z^2)}\left(\frac{s^2 + M_Z^4}{\sin^2\theta} - \frac{(s - M_Z^2)^2}{2}\right) \tag{2.6}$$

The cuts for the experimental data to be used, i.e. the range of θ in that the detector is sensitive and over which we have to integrate, are given in two ways: In the form $x_min < \cos\theta < x_max$ and in the form $p_\perp^\gamma > f_c\sqrt{s}/2$, as parameter f_c, where $p_\perp^\gamma$ is the photon impulse component transversal to the collider axis. For given s, this is equivalent to

$$|\sin\theta| > \frac{2f_c s}{|s - M_Z^2|}$$

and can be combined with the former cut during integration.

In general the energies of the outgoing particles in a two to two process like $e^+e^- \to \gamma Z$ are normally fixed by kinetics. The narrow-width approximation in the calculation of $|\mathcal{M}|^2$, that is the assumption that the Z is not virtual, fixes the photon energy for given (s, θ), therefore there is no p^γ dependency in (2.6). This also means that for $s < M_Z^2$, the process cannot take place. Therefore, in fig. 2.3 the dashed cross section is set zero for $\sqrt{s} < M_Z$, although this is not seen in (2.6). However because the Z boson quickly disintegrates it does not have to be on-shell. In that case the energy of the visible photon may vary.

If we consider the process $e^+e^- \to \gamma\nu\bar{\nu}$, there are also Feynman diagrams with $W^\pm$ bosons; however we will neglect those. Our method to calculate the $e^+e^- \to \gamma\nu\bar{\nu}$ cross section from $e^+e^- \to \gamma Z$ is called the narrow-width-approximation: we simply multiply the latter section by the "branching rate to invisible" of the Z boson, which is 20 %. [4] gives a more complete result for the examined cross section, obtained whithout using the narrow-width approximation, but still neglecting the contribution of the $W^\pm$ and not yet considering radiative corrections. Radiative corrections are often made because electrons (as all charged, accellerated particles) may already strongly radiate before the reaction takes place, thus enlargening the measured signal. We take the formulae of [4] as comparison:

$$e^+(p_+) + e^-(p_-) \to \bar{\nu}(q_+) + \nu(q_-) + \gamma(k) \tag{2.7}$$

$$\frac{d\sigma}{d\cos\theta dk} = \frac{\alpha}{12\pi^2} G^2 M_W^4 \frac{s'k}{s\kappa_+\kappa_-} \left[\eta_+^2 F(\eta_+) + \eta_-^2 F(\eta_-)\right] \tag{2.8}$$

with the G being the Fermi constant, N_ν the number of neutrino generations (=3) and

$$\eta_\pm = \frac{s - \kappa_\pm}{M_W^2},\ \kappa_\pm = 2p_\pm k,\ s' = (q_+ + q_-)^2, Z = s' - M_Z^2 + iM_Z\Gamma_Z \tag{2.9}$$

$$
\begin{aligned}
F(\eta_\pm) &= N_\nu \frac{1}{2}\left((g_v+g_a)^2+(g_v-g_a)^2\right)\frac{M_Z^4}{|Z|^2} \\
&\quad +3(g_v+g_a)\frac{M_Z^2 \mathrm{Re} Z}{|Z|^2}\frac{1}{\eta_\pm}\left(3+\frac{2}{\eta_\pm}-2\left(1+\frac{1}{\eta_\pm}\right)^2\ln(1+\eta_\pm)\right) \\
&\quad +\frac{6}{\eta_\pm^2}\left((1+\eta_\pm)\left(1-\frac{2}{\eta_\pm}\ln(1+\eta_\pm)\right)+1\right) \qquad (2.10)
\end{aligned}
$$

This cross section is numerically integrated for $1\,\mathrm{GeV} < |\vec{k}| < 100\,\mathrm{GeV}$ and also shown in fig. 2.3. Here, a non-zero cross section at $\sqrt{s} < M_Z$ is allowed.

Comparision with experimental data from ALEPH at LEP, CERN [6] (table 2.1) shows neither curve fits well (fig. 2.3). The dashed curve fits if multiplied by a factor 2.75, the continous does not even fit if scaled.

All calculations were done using a running α coupling constant as shown in fig. 2.4 but taking fixed values for all other couplings and masses. However this does not change much the quality of the fit.

The cut condition $p_\perp^\gamma < 0.0375\sqrt{s}/2$ has some interesting effects in the region of ca. 80 GeV – 104 GeV (see fig. 2.7), but is covered by the condition $|\cos\theta| < 0.95$ in the region of the data points.

Clearly considering the left-out $W^\pm$ contribution or radiative corrections would both increase the seen cross section, however this is already too high. Considering the $W^\pm$ would not increase σ by a significant factor, but considering radiative corrections could.

$\sqrt{s}$ (GeV)	σ (pb)	total error (pb)
189	3.43	0.16
192	3.47	0.39
196	3.03	0.22
200	3.23	0.21
202	2.99	0.29
205	2.84	0.21
207	2.67	0.16

Table 2.1: experimental data for single photon events plus missing energy from ALEP at LEP, CERN ([6]), for $p_\perp^\gamma < 0.0375\sqrt{s}/2$ and $|\cos\theta| < 0.95$.

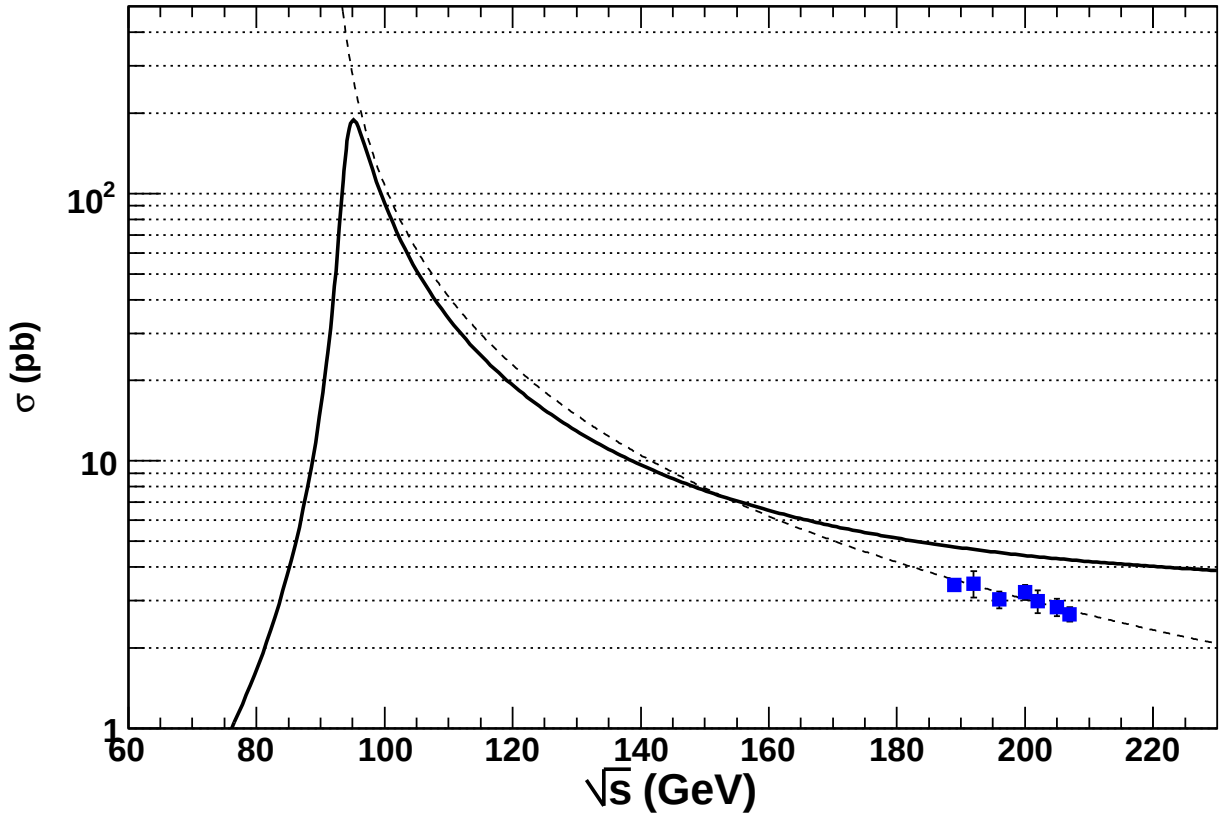

Figure 2.3: cross section calculated with narrow-width approximation. The curve is normed by 1/2.75 so it goes through the data. Continuous line: cross section without narrow-width approximation, but still neglecting $W^{\pm}$ contributions. Both curves are calculated only with the cut $|\cos\theta| < 0.95$

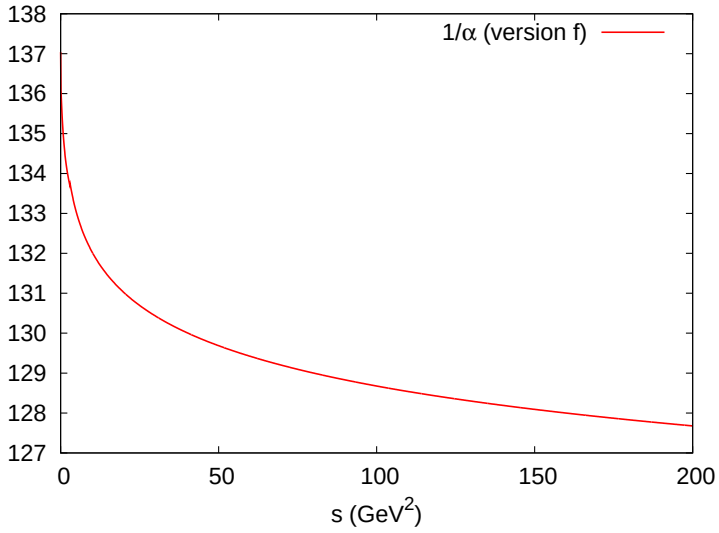

Figure 2.4: running α function used in all calculations

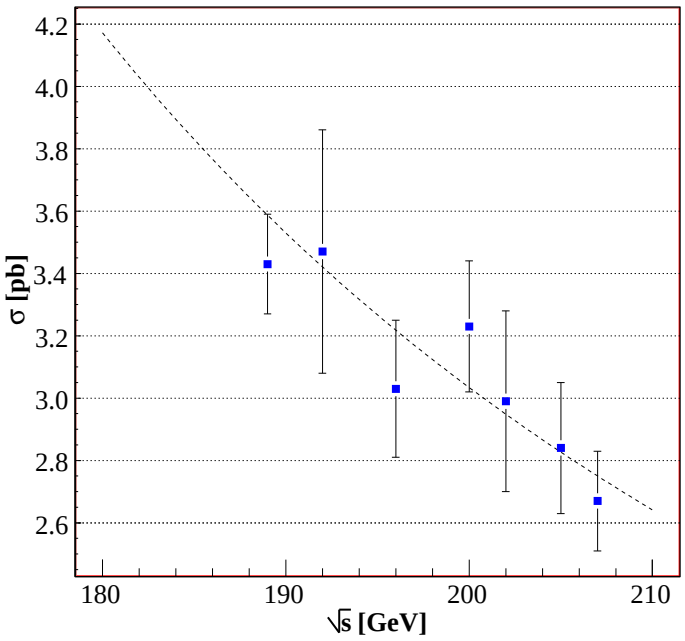

Figure 2.5: dashed line: cross section calculated with narrow-width approximation (see eq. 2.6). The curve is normed by the factor 1/2.75 so it goes through the data points. Cut condition is $|\cos\theta| < 0.95$

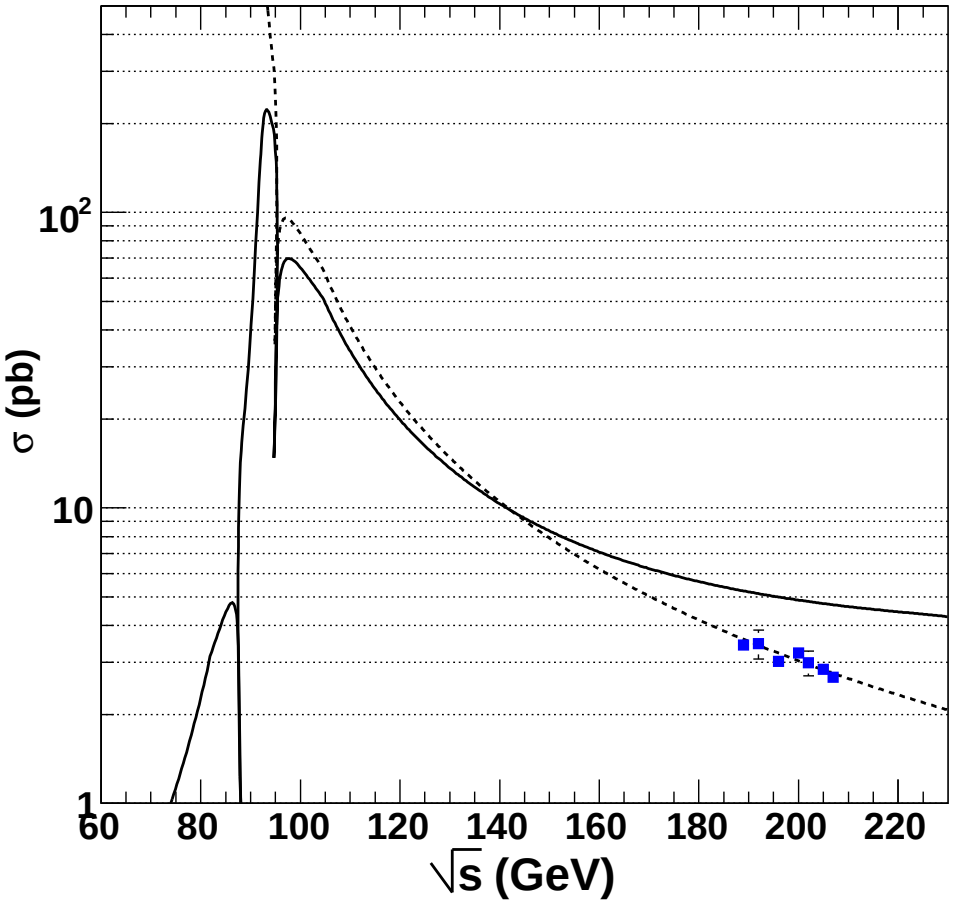

Figure 2.6: settings like fig. 2.5 but also with cut $p_\perp^\gamma < 0.0375\sqrt{s}/2$. Compare fig. 2.7

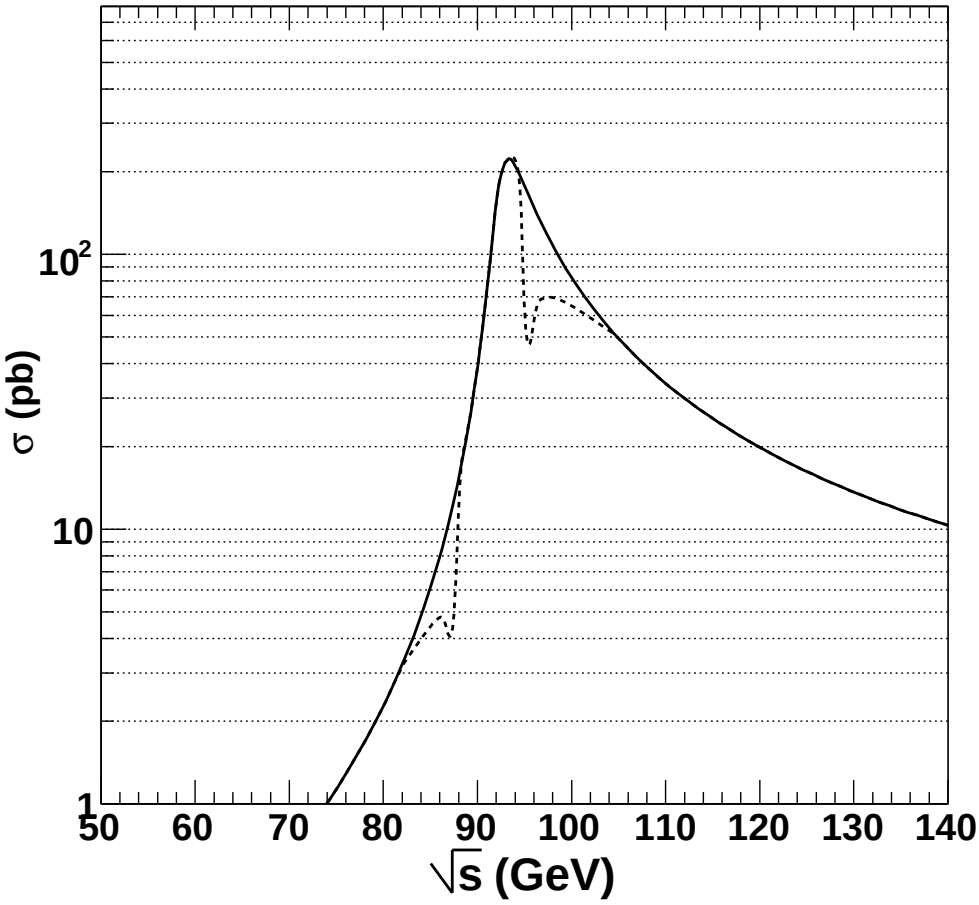

Figure 2.7: The effect of the additional cut $p_\perp^\gamma < 0.0375\sqrt{s}/2$: The continuous curve is calculated only with the cut $|\cos\theta| < 0.95$, the dashed one with both cuts. The other parameters are as in fig. 2.5, continuous curve.

2.2 Numerical Integration with Monte Carlo/Vegas

The idea of Monte Carlo integration, is to calculate a d-dimensional integral by approximating the d+1-dimensinal volume enclosed by the integrand function graph. This can be done by choosing random points in an enclosing domain of the function graph and then testing if they are inside the volume of the graph. An equivalent, but much faster method is calculation of the mean function value of the points. Advanced implementions like the Vegas method of the CUBA library ([7]), which I used, use quasi-random numbers instead of pseudo-random numbers. The term pseudo-random stands for computer generated "random" numbers (non-periodic & equally distributed, but not truly random). This method is quite slow (convergence rate $O(1/\sqrt(n))$, where n is the number of evaluations).

Choosing numbers from a fixed grid makes convergence much quicker ($O(\frac{\log^d n}{n})$), but a high-dimensional grid cannot be easily refined during integration. Quasi-random numbers try to fill the gap: they behave randomly in a global way, but at the same time try to avoid areas where point density is already high. Monto-carlo based integration methods are especially useful when trying to handle bad-behaving or statistical functions, where classical taylor series based integration methods do not perform well.

2.3 Further Proceeding

In order to be able to extract contraints for a Z', first the data must be quite precisely reproducable by means of the standard model. Therefore we will have to further investigate for errors or unmatched assumptions in this so-far description. The original idea was that we would perhaps already achieve high precision in the relevant area of energy already with the simple formula (2.6). When the problem of the current calculation will be known, we will again be able to test this approach. The final aim is a exclusion plot in Z' parameters, probably done by the χ^2 method which can give an exclusion probability for each tested point in the space of Z' couplings and masses.

Bibliography

[1] Francis Halzen and Alan D. Martin, "Quarks & Leptons: An Introductory Course in Modern Particle Physics", John Wiley & Sons, 1984

[2] Course of 2006-2007: "Physique des Particules" at Université de la Méditerranée (Aix-Marseille II), http://marwww.in2p3.fr/∼talby/Phys_Particules/M2/

[3] Pierre Fayet, "U-boson production in e^+e^- annihilations, ψ and Υ decays, and Light Dark Matter, arXiv:hep-ph/070217v1, 17 Feb 2007

[4] F.A. Berends, G.J.H. Burgers, C. Mana, M. Martinez and W.L. van Neerven, "Radiative Corrections to the Process $e^+e^- \to \nu\bar{\nu}\gamma$, Nuclear Physics B301 (1988) p. 583

[5] A. Leike, "The Penomenology of Extra Neutral Gauge Bosons", arXiv:hep-ph/9805494v1, 28 May 1998

[6] ALEPH at CERN, "Single- and multi-photon production in e^+e^- collisions at $\sqrt{s}$ up to 209 GeV", European Physics Journal C 28 (2003) p. 1

[7] T. Hahn, "Cuba - a library for multidimensional numerical integration", arXiv:hep-ph/0404043 v2 26 Jan 2005

[8] M. Jamin and M.E. Lautenbacher, Tracer package and manual, http://library.wolfram.com/infocenter/MathSource/2987/

[9] Root Data Analysis Framework, http://root.cern.ch/

Appendix A

Program Codes

Here I include some of my programs/codes, only selecting those with more unique features or applications.

A.1 Tracer Code

```
<< tracer.m
VectorDimension[4]
AntiCommute[on]

(* e+,e- -> gamma,Z *)
OnShell[on,  {k1, 0}, {k2, Mzsq}, {p1, 0}, {p2, 0},
   {k1, k2, (s-Mzsq)/2},
   {p1, p2, s/2},
   {p1, k1, -t/2},
   {p2, k2, -(t-Mzsq)/2},
   {p1, k2, -1/2 (u - Mzsq)},
   {k1, p2, -1/2 u} ]

Spur[m1sq, m2sq, m1m2]
(* Minv = 1/M^2 *)
Gm1sq = G[m1sq, p2, V U + A G5, {nu2}, k1-p1, p1, k1-p1, {nu}, V U - A G5]

M1sq = FullSimplify[ -2 ({nu2}.{nu} - Minv k2.{nu2} k2.{nu}) Gm1sq e^2 /(4t^2) ]

Gm2sq = G[m2sq, p1, {mu}, V U - A G5, k2 - p1, p2, k2 - p1, V U + A G5, {mu2}]

M2sq = FullSimplify[ -2 ({mu2}.{mu} - Minv k2.{mu2} k2.{mu})
```

```
                    * Gm2sq e^2 / (4 u^2) ]

Gm1m2 = G[m1m2, p1, {mu}, k1-p1, {nu}, V U - A G5, p2, {mu},
          k2-p1, V U + A G5, {nu2}]

M1M2 = FullSimplify[ ({nu2}.{nu} - Minv k2.{nu2} k2.{nu}) Gm1m2 e^2 / (4 t u) ]
M1M2 = M1M2 /. s+t+u->Mzsq

Print[ "M1M2 = ", M1M2 ]
Print[ "with {V->1, A->0, Minv->0}: M1M2 = ",
   M1M2 /. {V->1, A->0, Minv->0}]

Ma = FullSimplify[ M1sq+M2sq + 2 ComplexExpand[Re[M1M2]] ]
Mb = Ma /. {s->Mzsq-t-u, Minv->1/Mzsq}
Mc = Mb /. {t->1/2 (s-Mzsq)(Cos[theta]-1), u->1/2 (Mzsq-s)(Cos[theta]+1)}
Md = FullSimplify[Mc 1/16 alpha/e^2 (s-Mzsq)/s^2]

sigmaMandelstam = FullSimplify[Mb]
sigmaTheta = Md
```

A.2 ROOT and CUBA usage

A program that numerically integrates the formulas by F.A. Berends using CUBA/Vegas, is also linked against ROOT librarys to plot results

```
// general includes
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <getopt.h>
#include <fstream>
#include <iostream>
using namespace std;

#include "cuba.h"
#include "alpha.h"

// ROOT includes
#include "TApplication.h"
#include "TROOT.h"
#include "TCanvas.h"
#include "TGraph.h"
```

```
#include "TMultiGraph.h"
#include "TLegend.h"
#include "TStyle.h"
#include "TMarker.h"
#include "TPaveText.h"
#include "TPaveLabel.h"
#include "TGraphErrors.h"
#include "TLatex.h"
#include "TString.h"
#include "TFrame.h"


// #define COMPARE_BERENDS
#define COMPARE_LEP2

// CUBA Vegas integration parameters
#define EPSREL 1e-3
#define EPSABS 1e-12
#define VERBOSE 0

#define MINEVAL 0
#define MAXEVAL 50000

#define NSTART 1000
#define NINCREASE 500

#define PR(x) cout << #x "= " << x << endl;

static inline double Sq(double x) {
   return x*x;
}

const double alpha_0 = 1./137.03604;
#ifdef COMPARE_BERENDS
// values taken from paper "radiative corrections" by F.A. Berends et al.
const double M_W = 82;
const double M_Z = 93;
const double Gamma_Z = 2.7;
#endif
#ifdef COMPARE_LEP2
const double M_W = 80.398;
const double M_Z = 91.1876;
const double Gamma_Z = 2.4952;
```

```
#endif

const double sinThetaW = sqrt(1 - Sq(M_W/M_Z));
const double e = sqrt(4*M_PI*alpha_0);
const double g = e / sinThetaW;
const double G = sqrt(2.)*Sq(g/M_W)/8;
const double gv = -1./2 + 2*Sq(sinThetaW);
const double ga = -1./2;
const int N_nu = 3;

const double gev2pbarn = 0.389379324e9; // conversion from GeV^-2 to pbarn

double s;

static double FOfEta( double eta, double s_apostr) {
   double Z_Real = s_apostr - Sq(M_Z);
   double Z_Im = M_Z*Gamma_Z;
   double Z_AbsSq = Sq(Z_Real) + Sq(Z_Im);

   return N_nu/2. * (Sq(gv+ga) + Sq(gv-ga)) * pow(M_Z, 4)/Z_AbsSq
      + 3.*(gv+ga)* Sq(M_Z)*Z_Real/Z_AbsSq /eta
      * (3 + 2./eta - 2*Sq(1+1/eta)*log(1+eta))
      + 6./Sq(eta)* ((1+eta)*(1-2./eta*log(1+eta)) + 1);
}

// Integrand. Will be integrated in area [0,1]x[0,1].
static void sigmaDCosDK( const int *ndim, const double xx[],
                         const int *ncomp, double ff[]) {
   // integration result will also have to be multiplied
   // by scale applied on arguments
   double scaleResult = 1;

   // rescale: we want cos(theta) in [-1,1]
   double costheta = xx[0]*2 - 1;
   scaleResult *= 2;

   // rescale: we want k in [kmin,kmax] GeV
   double kmin = 1, kmax = 100;
   double dk = kmax-kmin;
   double k = kmin + dk*xx[1];
   scaleResult *= dk;

   // put cuts here:
```

```
#ifdef COMPARE_BERENDS
   // general cut at both ends
   double cutdegree = 20;
   // p_transversal > cut_ptrans*sqrt(s)
   double cut_ptrans = 0;
#endif
#ifdef COMPARE_LEP2
   double cutdegree = 180/M_PI*acos(0.95);
   double cut_ptrans = 0.0375;
#endif

   double cutcos_ptrans = cos(asin(2*s*cut_ptrans/fabs(s-M_Z*M_Z)));
   // additional condition is now |cos(theta)|<cutcos_ptrans

   double cutcos = cos(cutdegree*M_PI/180);
   if (cutcos > cutcos_ptrans) {
      cutcos = cutcos_ptrans;
   }
   if ( costheta < -cutcos ||  costheta > cutcos  ) {
      ff[0] = 0;
      return;
   }

   // s is given as global variable
   double E = sqrt(s)/2;

   // p+,p- and K are 4-vectors
   // p+ = (E, 0, 0, E), p- = (E, 0, 0, -E),
   // K = (k, k sin(theta), 0, k cos(theta))
   // k+/- = 2p+/- K
   double k_plus = 2*(E*k+E*k*costheta);
   double k_minus = 2*(E*k-E*k*costheta);

   // s' = (q+ + q-)^2 = (p+ + p- - K)^2
   double s_apostr = Sq(2*E - k) - Sq(k);

   double eta_plus = (s-k_plus)/Sq(M_W);
   double eta_minus = (s-k_minus)/Sq(M_W);

   double integral = alpha(s)/(12. *Sq(M_PI)) * Sq(G) * pow(M_W, 4)
      * s_apostr *k/(s*k_plus*k_minus)
      * (Sq(eta_plus)*FOfEta(eta_plus, s_apostr)
         + Sq(eta_minus)*FOfEta(eta_minus, s_apostr));
```

```
   ff[0] = scaleResult * integral * gev2pbarn;
}

// in mbarn
double sigmaTotal(double s_) {
   s = s_;
   const int ncomp = 1;
   const int ndim = 2;

   int comp, nregions, neval, fail;
   double integral[ncomp], error[ncomp], prob[ncomp];

   double result = integral[0];
   if (VERBOSE >= 0) {
      printf("sigma(%.8f) = (%.8f +- %.8f) pbarn, probchi=%f, fail=%d\n",
             s, result, error[0], fail, prob[0]);
   }
   return result;
}

int main( int argc, char* argv[] ) {
   // integrate sigma values
   double xmin = 60, xmax = 160;  // may be overwritten by command line
   int nbpoints = 100;

   // Root initialization; this cuts away Root command line arguments
   TApplication app("plot", &argc, argv);

   const char shopt[] = "cp";
   const struct option lopt[] = {
      // only compute and save data
      { "compute-only", no_argument, NULL, 'c' },
      // only take computed data and plot
      { "plot-only", no_argument, NULL, 'p' },
      { "ssqrmin",           required_argument, NULL, '0' },
      { "ssqrmax",           required_argument, NULL, '1' },
      { "steps",             required_argument, NULL, 'n' },
      { 0, 0, 0, 0 }
   };

   bool compute = true;
   bool doPlot = true;
```

```
for(int c = getopt_long(argc, argv, shopt, lopt, NULL);
    c != -1;
    c = getopt_long(argc, argv, shopt, lopt, NULL))  {
   switch(c) {
   case 'c':
      doPlot = false;
      break;
   case 'p':
      compute = false;
      break;
   case '0': xmin = atof(optarg); break;
   case '1': xmax = atof(optarg); break;
   case 'n': nbpoints = atoi(optarg); break;
   }
}

double xvals[nbpoints], sigvals[nbpoints];

// opening read+write doesn't work (don't know why)
ios_base::openmode mode = compute ? fstream::out : fstream::in;
fstream datfile("singlephoton-graph.dat",  mode);
if (!datfile.is_open())
   cerr << "problem opening data file" << endl;

for (int i=0; i<nbpoints; i++) {
   if (compute) {
      xvals[i] = xmin + (xmax-xmin)*double(i)/(nbpoints-1);
      sigvals[i] = sigmaTotal( Sq(xvals[i]));

      datfile << xvals[i] << "\t" << sigvals[i] << endl;
      if (!datfile)
         cerr << "problem writing to data file" << endl;
   }
   else {
      datfile >> xvals[i] >> sigvals[i];
      if (!datfile)
         cerr << "problem reading data file" << endl;
   }
}
datfile.close();

if (!doPlot)
   return 0;
```

```
    TCanvas *plot = new TCanvas("plot", "plot", 200, 10, 500, 800);
    plot->SetGridy();
    plot->SetGridy();
    plot->SetLogy();
    plot->SetFillColor(10);
    plot->SetBorderMode(0);
    plot->SetTickx(1);

    TGraph *graph = new TGraph(nbpoints, xvals, sigvals);
    graph->SetTitle("numerical integrated e+ e- -> #gamma#nu#bar{#nu}");
    graph->Draw("AC");

    // set line options
    graph->SetLineColor(1);
    graph->SetLineWidth(2);
    graph->SetLineStyle(1);

    // set axis titles
    TAxis* xAxis = graph->GetXaxis();
    xAxis->SetLimits(50, 160);
    xAxis->SetTitle("#sqrt{s} (GeV)");
    xAxis->CenterTitle(1);
    xAxis->SetTitleOffset(0.85);
    xAxis->SetTitleSize(0.05);

    TAxis* yAxis = graph->GetYaxis();
    // yAxis->SetLimits(1, 500); // doesn't work in log mode
    yAxis->SetTitle("#sigma (pb)");
    yAxis->CenterTitle(1);
    yAxis->SetTitleOffset(1.2);
    yAxis->SetTitleSize(0.04);


    graph->SetMinimum(1);
    graph->SetMaximum(800);
    plot->SaveAs("singlephoton-graph.eps");


    app.Run();
    return 0;
}
```

Evaluation of narrow-width σ forumla

```
#include <getopt.h>
#include <cmath>
#include <iostream>
using namespace std;

#include "alpha.h"

const double gev2pbarn = 0.389379324e9; // conversion from GeV^-2 to pbarn
const double br = 0.2; // fractional branching ratio for Z->invisible

// this is the unspecified integral of dsigma/dcos(theta), where x = cos(theta)
double sigint_unsp(double s, double x, double gv, double ga, double mz) {
   return alpha(s) * (ga*ga+ gv*gv) / (2* s*s *fabs(s - mz*mz)) *
      ( -pow(s - mz*mz, 2)/2. * x + ((pow(mz,4) + s*s))
        * log(fabs((1 + x)/(-1 + x)))/2.);
}

// total cross section of e+e- -> gamma,Z process (onshell) with
// coupling constants ga,gv
// mass of Z mz and cuts costhmin, costhmax
// cut_ptrans: for cut  p_transversal>cut_ptrans*sqrt(s)
// cross section in GeV^-2
double sigtot (double s, double costhmin, double costhmax, double cut_ptrans,
               double gv, double ga, double mz) {
   double cutcos_ptrans = cos(asin(2*s*cut_ptrans/fabs(s-mz*mz)));
   // additional condition is now |cos(theta)|<cutcos_ptrans
   if (costhmax > cutcos_ptrans)
      costhmax = cutcos_ptrans;
   if (costhmin < -cutcos_ptrans)
      costhmin = -cutcos_ptrans;
   // if there is not enough energy to create a Z boson
   // our on-shell gamma,Z cross section must be zero,
   // even if the formula doesn't (yet?) reflect this
   if (s<mz*mz)
      return 0;
   return sigint_unsp(s, costhmax, gv, ga, mz)
      - sigint_unsp(s, costhmin, gv, ga, mz);
}
int main( int argc, char *argv[]) {
   double costhmin = -0.95;  // may be overwritten by command line
   double costhmax = 0.95;   //
```

```
   double cut_ptrans = 0.0375;  // ## p_transversal > cut_ptrans*sqrt(s)
   double ssqr_min = 180;
   double ssqr_max = 210;
   double h = 100;

   const char shopt[] = "l:u:";
   const struct option lopt[] = {
      { "lower-costheta", required_argument, NULL, 'l' },
      { "upper-costheta", required_argument, NULL, 'u' },
      { "cut_ptrans",     required_argument, NULL, 't' },
      { "ssqrmin",           required_argument, NULL, '0' },
      { "ssqrmax",           required_argument, NULL, '1' },
      { "steps",             required_argument, NULL, 'n' },
      { 0, 0, 0, 0 }
   };
   for(int c = getopt_long(argc, argv, shopt, lopt, NULL);
       c != -1;
       c = getopt_long(argc, argv, shopt, lopt, NULL))  {
      switch(c) {
      case 'l': costhmin = atof(optarg); break;
      case 'u': costhmax = atof(optarg); break;
      case 't': cut_ptrans = atof(optarg); break;
      case '0': ssqr_min = atof(optarg); break;
      case '1': ssqr_max = atof(optarg); break;
      case 'n': h = atoi(optarg); break;
      }
   }
   // Z parameters
   double ga = -0.50123;
   double gv = -0.03783;
   double mz = 91.1876;

   double step = (ssqr_max - ssqr_min)/h;
   for (int i=0; i<=h; i++) {
      double ssqr = ssqr_min + i*step;

      double s = ssqr*ssqr;
      double sig = gev2pbarn*br*
         sigtot(s, costhmin, costhmax, cut_ptrans, gv, ga, mz);
      cout << ssqr << "\t" << sig << endl;
   }
   return 0;
}
```