Suche nach transienten Ereignissen in Fermi-LAT Beobachtungen

Searching for transient events in Fermi-LAT Data

von
Alicia Hirt
geboren am
17. Januar 1989

Bachelor-Arbeit im Studiengang Physik
Universität Hamburg
03.04.2012

1. Gutachter: Prof. Dr. Dieter Horns
2. Gutachter: Prof. Dr. Robi Banerjee

Zusammenfassung

Ziel dieser Arbeit ist es, mittels der Suche nach transienten Ereignissen eine Grenze auf die Massendichte ρ_{PBH} primordialer Schwarzer Löcher zu setzen. Dazu wurden die Daten des Fermi-LAT verwendet und 100 s lange Zeitfenster auf das Auftreten vieler hochenergetischer Photonen untersucht. Die Umsetzung ist in dieser Arbeit dokumentiert. Anhand mehrerer Gamma-ray Bursts wird gezeigt, dass die Methode funktioniert. Allerdings werden nur 2000 von 12288 Pixeln, in die der Himmel unterteilt wurde, untersucht. Abschließend wird eine vorläufige Grenze auf die Massendichte ρ_{PBH} gesetzt: $\rho_{\mathrm{PBH}} \lesssim 3{,}7 \cdot 10^{21} \frac{\mathrm{g}}{\mathrm{pc}^3}$.

Abstract

Object of this thesis is the evaluation of an estimate on the mass density ρ_{PBH} of evaporating primordial black holes by searching for transient events. Therefore data from the Fermi-LAT has been used and 100 s long time slots have been investigated for the arrival of many high enegetic photons. The realisation is documented in this thesis. By the use of several Gamma-ray Bursts it is shown that this analysis is working. Nevertheless only 2000 of the 12288 pixels the sky was diveded into have been analysed. Finally a provisional limit on the mass density ρ_{PBH} is set: $\rho_{\mathrm{PBH}} \lesssim 3{,}7 \cdot 10^{21} \frac{\mathrm{g}}{\mathrm{pc}^3}$.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Schon seit mehreren Jahrtausenden beschäftigt sich die Menschheit mit der Astronomie. Bereits etwa 2700 v.Chr. erhielten in Babylon viele der nördlichen Sternbilder ihre Namen. Lange Zeit fanden diese Beobachtungen ausschließlich im optischen Bereich statt. Im 19. Jahrhundert konnten dank Spektralanalyse schon Aussagen über die chemische Zusammensetzung von Sternen getroffen werden. Im Jahr 1990 startete mit dem *Hubble Space Telescope* ein extraterrestrisches Großteleskop, welches der Menschheit bis heute viele Erkenntnisse über das Weltall außerhalb unseres Sonnensystems bringt. Mitte des 20. Jahrhunderts ermöglichte der Fortschritt der Technik Beobachtungen auch in anderen Wellenlängenbereichen. So entdeckte 1967 ein Vela-Satellit, auf der Suche nach einem Verstoß gegen das Atomwaffenverbot, durch Zufall erstmals einen Gammastrahlenausbruch (engl. Gamma-ray Burst, GRB), welcher unter die Kategorie „transientes Ereignis“ fällt. 1968 wurde der erste Pulsar gefunden. Die Forschung in Wellenlängenbereichen jenseits des Optischen wurde mit vielen Satelliten vorangetrieben. So deckte beispielsweise das Experimet EGRET (*Energetic Gamma-Ray Experiment Telescope*) ein Spektrum von 20 MeV bis ca. 20 GeV ab. Mit jeder Entdeckung eines bisher unbekannten Phänomens werden neue Fragen aufgeworfen. Im Juni 2008 startete das *Fermi Gamma-ray Space Telescope* mit dem Ziel, viele der heute noch offenen Fragen zu beantworten.

Zielsetzung dieser Arbeit ist es, mit Hilfe der vom Fermi-Teleskop gesammelten Daten, neben Gamma-ray Bursts auch nach anderen transienten Ereignissen im Energiebereich von Gigaelektronenvolt (GeV) zu suchen. Ein möglicher Kandidat für ein transientes Ereignis ist ein evaporierendes primordiales Schwarzes Loch (engl.: primordial black hole, PBH), welches innerhalb kürzester Zeit eine große Menge Energie freisetzt. Die Daten des Fermi-Teleskops werden auf die Ankunft vieler energiereicher Photonen innerhalb kurzer Zeiträume untersucht. Die Koordinaten möglicher „PBH“-Kandidaten werden mit Daten bereits bekannter Gammaquellen oder GRBs verglichen. Schlussendlich wird eine obere Grenze auf die Anzahl von primordialen Schwarzen Löchern pro Volumen gesetzt.

Die Arbeit ist in mehrere Kapitel gegliedert.
Nach einigen einleitenden Sätzen in Kapitel 1 wird in Kapitel 2 für die in dieser Arbeit durchgeführte Analyse wichtiges Hintergrundwissen erläutert. Es wird auf den Begriff des transienten Ereignisses im Allgemeinen und auf Gamma-ray Bursts und Schwarze Löcher im Speziellen eingegangen. Desweiteren folgt ein Überblick über das Fermi Gamma-ray Space Telescope (Fermi). Die frei erhältliche Software zur Auswertung von Daten des Fermi-Teleskops wird näher erläutert, genau wie auf das Programm HEALPix, zur Unterteilung einer Kugeloberfläche in einzelne Pixel.
In Kapitel 3 wird zunächst auf den verwendeten Datensatz und die daran vollzogenen Schnitte eingegangen. Anschließend wird die Methodik der Analyse dieser Arbeit im Detail vorgestellt. Sämtliche zur Auswertung verwendeten Programme werden vorgestellt und die Probleme erläutert, die während der Entwicklung entstanden sind.
Daraufhin folgen in Kapitel 4 die Ergebnisse der Analyse. Die Methode wird anhand von Gamma-ray Bursts auf ihre Richtigkeit überprüft, Hintergrundpixel sowie weitere helle Quellen werden erläutert und anschließend eine Himmelskarte mit den soweit ausgewerteten Pixeln gezeigt.
Das Kapitel 5 beschäftigt sich mit der Auswertung der Ergebnisse und der Berechnung einer Grenze auf die Massendichte primordialer Schwarzer Löcher.
Abschließend werden in Kapitel 6 die gewonnenen Erkenntnisse zusammengefasst und in den Kontext künftiger Analysen gesetzt.

Kapitel 2

Theoretischer Hintergrund

2.1 Transiente Ereignisse

Astrophysikalische Ereignisse können ein sehr unterschiedliches zeitliches Verhalten aufweisen. Liegt der Fokus auf einer Variation in der Helligkeit, so kann zwischen periodischen, quasi-periodischen, transienten und persistenten Ereignissen unterschieden werden. Die Helligkeit persistenter Phänomene ist zeitlich konstant, die von quasi-periodischen hingegen variiert. In kleinen Zeitintervallen können Perioden ausgemacht werden, auf größeren Zeitskalen hingegen ist keine Regelmäßigkeit erkennbar. Während periodische Ereignisse regelmäßig wiederkehren und ihnen eine konstante Periodendauer zugeordnet werden kann, sind transiente Ereignisse unregelmäßig und weisen keine zuordenbare Periodendauer auf (Müller, 2007). Transiente Ereignisse sind in der Regel von kurzer Dauer[1], können jedoch bis zu mehreren Tagen oder gar Monaten andauern (Keane, 2011, S.1). Typische Beispiele für transiente Ereignisse sind Gamma-ray Bursts und Supernovae. Ursachen solcher Phänomene können jedoch auch zum Beispiel die Sonne, Braune Zwerge oder evaporierende primordiale Schwarze Löcher sein (Keane, 2011, Kapitel 1.3).

Im Folgenden soll auf Gamma-ray Bursts und primordiale Schwarze Löcher näher eingegangen werden, da dies die bedeutendsten Kandidaten für diese Arbeit sind. In beiden Fällen handelt es sich um Phänomene, bei denen in kurzer Zeit viele Photonen detektiert werden können.

2.1.1 Gamma-ray Bursts

Gamma-ray Burts sind kurze, helle Explosionen im Universum. Die von ihnen in kürzester Zeit (Millisekunden bis Minuten) abgestrahlte Energie kann mit der bei einer Supernova-Explosion über mehrere Monate frei werdenden Energie verglichen werden. Die Energie eines aus einem Gamma-ray Burst stammenden Photons liegt in der Regel im Kiloelektronenvolt (keV) bis Megaelektronenvolt (MeV) Bereich. Es wurden jedoch auch schon Gamma-ray Bursts mit Photonen von mehreren GeVdetektiert. Gamma-ray Bursts sind heutzutage bis in große

[1] Besonders kurzzeitige Ereignisse haben eine Dauer von einigen Nanosekunden (ns) (Keane, 2011, S.1).

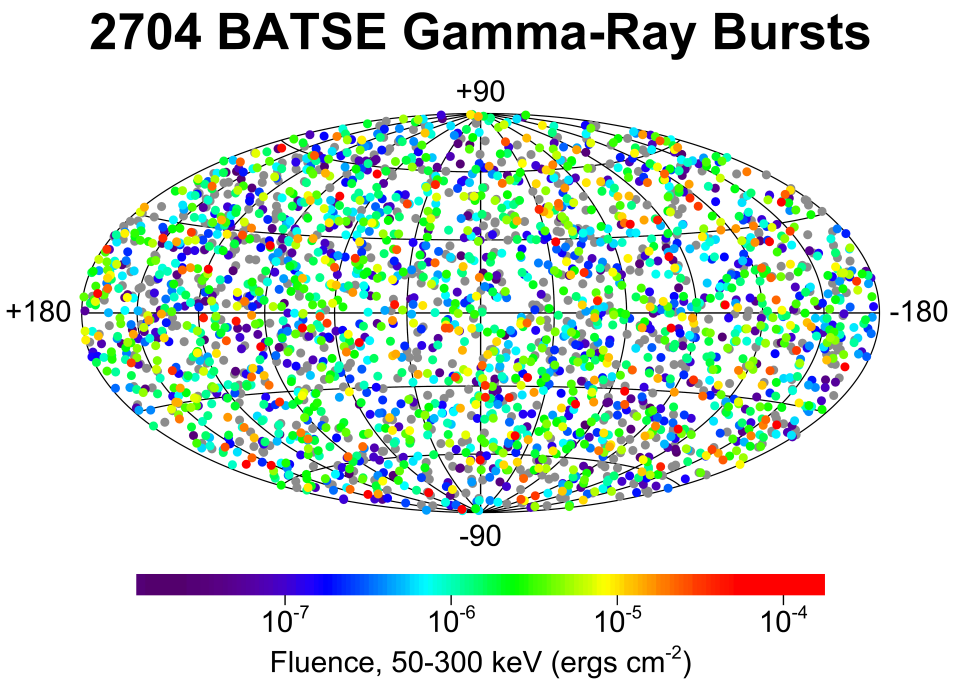


Abbildung 2.1: Himmelskarte in galaktischen Koordinaten mit den 2704 von BATSE detektierten GRBs. Fluenz ist als Energie pro Fläche definiert. (Michael S. Briggs, 2000)

galaktische Entfernungen mit Rotverschiebungen von $z \simeq 10$ messbar. Mit dem *BATSE*-Experiment (Burst And Transient Source Explorer) wurden von 1991 bis 2000 insgesamt 2704 Gamma-ray Bursts im Energiebereich von 50 keV bis 300 keV detektiert (Michael S. Briggs, 2000). Wie in Abbildung 2.1 zu sehen, wurde mit BATSE eine isotrope Verteilung der Gamma-ray Bursts im Himmel festgestellt (Bouvier, 2010). Je nach Fluenz sind die Gamma-ray Bursts in unterschiedlichen Farben markiert. Die Fluenz gibt die Energiemenge an, die pro Fläche detektiert wurde.

Trotz langjähriger Forschung sind die Ursachen von Gamma-ray Bursts immer noch unklar. Ein Modell ist das sogenannte *Feuerball*-Modell (engl. *fireball model*). Da dieses für diese Arbeit jedoch nicht von Bedeutung ist, wird im Folgenden nicht näher darauf eingegangen. Eine ausführliche Beschreibung ist in der Veröffentlichung von Mézáros (P. Mézáros, 2002) zu finden.

Es können zwei Arten von Gamma-ray Bursts unterschieden werden. Zum einen gibt es die kurzen Gamma-ray Bursts, die eine Dauer $T_{90} < 2\,s$ aufweisen, und zum anderen existieren die langen Gamma-ray Bursts mit einer Dauer $T_{90} > 2\,s$. Mit T_{90} wird die Zeitdauer bezeichnet, in der 90% aller Photonen gemessen werden. Bei beiden Kategorien kann zunächst eine kurzzeitige Explosion, die sogenannte *prompt emission*, festgestellt werden, gefolgt von einem Nachglühen, das als *afterglow* bezeichnet wird. Diese *afterglow*-Phase kann unter Umständen

mehrere Wochen andauern (Bouvier, 2010, Kapitel 1.2.4, S. 11ff).

Die Dauer der *prompt emission* eines langen Gamma-ray Bursts reicht von Millisekunden (ms) bis hin zu mehreren Minuten. Im Mittel kann von einer Dauer von etwa 30 s ausgegangen werden (Mitchell et al.).

Lange Gamma-ray Bursts werden vor allem in Spiralgalaxien mit einer überdurchschnittlichen Sternbildungsrate beobachtet. Vor allem nahe massereicher[2] Sterne treten Gamma-ray Bursts vermehrt auf. Bei einigen langen Gamma-ray Bursts zeigt ein Vergleich des optischen Spektrums des *afterglow* mit dem einer Supernova viele Übereinstimmungen. Dies führt dazu, dass die Entstehung eines langen Gamma-ray Bursts mit Supernovae in Zusammenhang gebracht wird. Da einige Supernovae jedoch mit dem Ende der Evolution massereicher Sterne verknüpft sind, kann auch zwischen Gamma-ray Bursts und dem dem Tod massereicher Sterne eine Verbindung hergestellt werden (S.E. Woosley, 2003). Im Speziellen ein Wolf-Rayet-Stern scheint ein guter Kandidat zu sein (Bouvier, 2010). Das sogenannte *Collapsar*-Modell beschreibt den Kollaps des Kerns eines solchen Wolf-Rayet Sterns. Mehr Informationen über das *Collapsar*-Modell enthält der Bericht von Woosley et al. (S.E. Woosley, 2003).

Im Gegensatz zu den langen Gamma-ray Bursts weisen kurze Gamma-ray Bursts eine mittlere Dauer von nur 300 ms auf (Mitchell et al.). Sie werden bevorzugt in Regionen mit einer geringen bzw. ohne Sternbildung beobachtet. Demzufolge werden sie mit alten Sternenformationen in Verbindung gebracht. Im Gegensatz zu den langen Bursts kann kein Zusammenhang zu Supernovae hergestellt werden. Stattdessen werden Zusammenstöße von Objekten, wie Neutronensternen oder Schwarzen Löchern, vermutet. Allerdings kommen auch andere Möglichkeiten in Frage. Hintergründe dazu sind unter anderem der Dissertation von Bouvier (Bouvier, 2010, Kapitel 1.1) zu entnehmen.

GeV-Photonen Das Fermi-Teleskop hat bereits mehrere Gamma-ray Bursts mit Photonen, deren Energie über 1 GeV lag, detektiert. Entsprechend der Zielsetzung dieser Arbeit, transiente Ereignisse mit Energien oberhalb von 1 GeV zu finden, sollten genau diese Gamma-ray Bursts in der Auswertung zu finden sein. In der Dissertation von Bouvier (Bouvier, 2010) ist eine Auflistung mit Gamma-ray Bursts, die bis Januar 2010 detektiert wurden und Photonenergien von mehr als 1 GeV aufweisen. Die vier Gamma-ray Bursts, denen mehrere Photonen mit Energien im GeV-Bereich zuzuordnen sind, sind in Tabelle 2.1 zusammengestellt.

Die Positionen der Gamma-ray Bursts im Himmel sind in Abbildung 2.2 dargestellt. Wird die Karte von oben nach unten, von links nach rechts gelesen, so sind die vier Gamma-ray Bursts in dieser Reihenfolge abgebildet: GRB090902B, GRB 080916C, GRB 090926A und GRB 090510.

[2] $M > 20 M_\odot$; $M_\odot$ entspricht der Masse der Sonne, $M_\odot \approx 2 \cdot 10^{30}$ kg

Tabelle 2.1: Übersicht über 4 vom LAT bis Januar 2010 detektierte Gamma-ray Bursts mit Photonenergien $> 1\,\mathrm{GeV}$; ΔT entspricht einer ungefähren Angabe der Dauer der *prompt emission*, z der rekonstruierten Rotverschiebung. E_{max} gibt die Energie des höchstenergetischsten Photons an. (Bouvier, 2010, S.91)

GRB	ΔT[s]	#Ereignisse $> 100\,\mathrm{MeV}$	# Ereignisse $> 1\,\mathrm{GeV}$	E_{max}[GeV]	z
080916C	∼80	145	13	∼14	4.3
090510	∼2	∼150	∼20	∼31	0,9
090902B	∼20	∼200	∼30	∼33	1,8
090926A	∼20	∼150	∼50	∼20	2,1

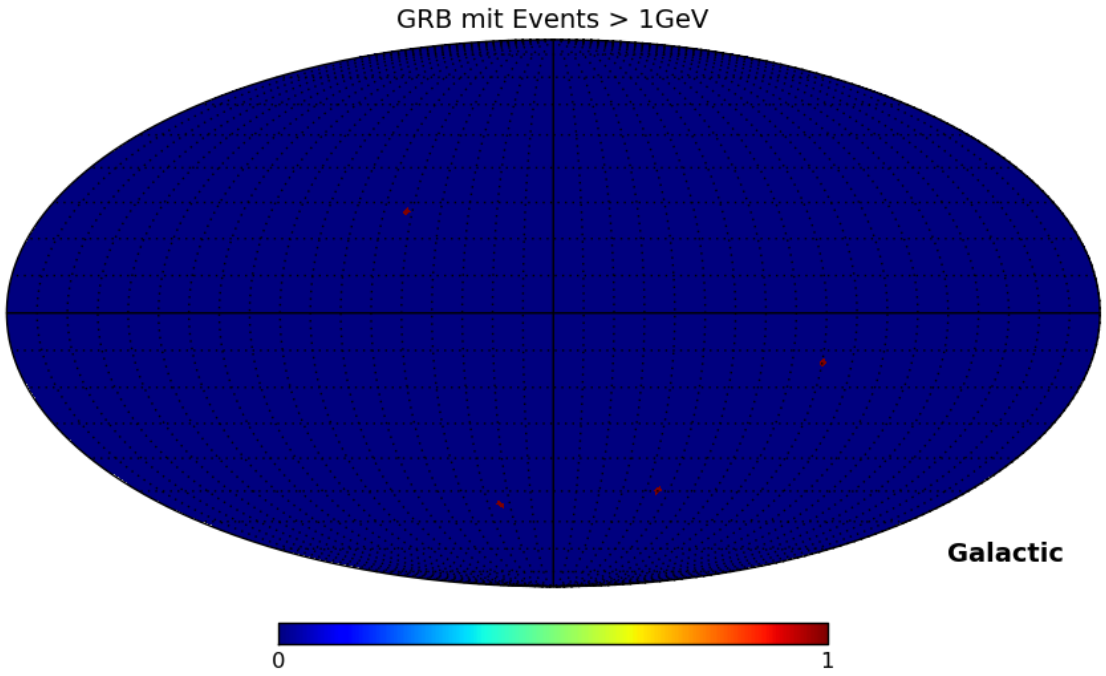


Abbildung 2.2: Himmelskarte in galaktischen Koordinaten mit den 4 in Tabelle 2.1 aufgelisteten Gamma-ray Bursts.

2.1.2 Schwarze Löcher

Als Schwarzes Loch bezeichnet man ein Objekt mit einer so hohen Dichte, dass selbst Licht dem gravitativen Einfluss nicht entkommt.
Die Idee, dass es ein solches Objekt geben könnte, stammt bereits aus dem Jahr 1783. John Michell beschrieb seine Idee 1783 vor der Royal Society wie folgt:

„*If the semi-diameter of a sphere of the same density as the Sun in the proportion of five hundred to one, and by supposing light to be attracted by the same force in proportion to its [mass] with other bodies, all light emitted from such a body would be made to return towards it, by its own proper gravity.*“

(John Michell an die Royal Society of London, 1783)

Die Massen Schwarzer Löcher können sich stark unterscheiden. Zum einen gibt es die sogenannten supermassiven Schwarzen Löcher, deren Massen mehreren Millionen Sternen [3] entsprechen. Sie werden in den Zentren großer Galaxien vermutet. Im Zentrum der Milchstraße befindet sich die Radioquelle Sagittarius A*, bei der es sich aller Voraussicht nach um ein supermassives Schwarzes Loch handelt. Trotz vieler Theorien ist die Entstehung dieser Objekte bisher ungeklärt (Harvard-Smithsonian Center for Astrophysics, 2008). Desweiteren existieren Schwarze Löcher, die wenige Sonnenmassen schwer sind. Sie entstehen durch den Gravitationskollaps eines Sterns. In der Endphase des Sterns kann der Strahlungsdruck nicht aufrecht erhalten werden. Der Gravitationsdruck überwiegt und der Stern kollabiert. Ist seine Masse größer als die sogenannte Chandrasekhar-Grenze, so entsteht ein Schwarzes Loch. Die Existenz Schwarzer Löcher mit einer geringeren Masse ist rein spekulativ, jedoch physikalisch erlaubt ('t Hooft, 2009, Kapitel 1, S.3). Auf diese sogenannten primordialen Schwarzen Löcher wird in Kapitel 2.1.3 näher eingegangen.

Bei der Beschreibung Schwarzer Löcher spielt das sogenannte „Keine-Haare-Theorem“ (engl.: no hair theorem) eine wichtige Rolle. Dieses besagt, dass stationäre Schwarze Löcher unter der Abwesenheit externer Felder — mit Ausnahme von dem elektromagnetischen Feld — durch maximal 3 Parameter beschrieben werden können. Dies sind die Masse M, der Drehimpuls J und die Ladung Q. Da jedes Schwarze Loch Masse besitzen muss, gibt es vier mögliche Kombinationen dieser Parameter (S. W. Hawking, 1994, Kapitel 2, S. 23). Diese sind in Tabelle 2.2 aufgelistet.

Die Schwarzschildlösung ist das einfachste Modell. Unter der Annahme, dass Ladung und Drehimpuls Null sind, kann der sogenannte Schwarzschildradius ausgerechnet werden. Unter Berücksichtigung der Newtonschen Gravitationstheorie ergeben sich die folgenden Überlegungen:

[3] $M_{SMBH} \approx 10^8 - 10^{10} M_\odot$

Tabelle 2.2: Verschiedene Modelle Schwarzer Löcher in Abhängigkeit der Parameter Masse M, Ladung Q und Drehimpuls J (S. W. Hawking, 1994).

Name	Masse	Ladung	Drehimpuls
Schwarzschild Lösung	M	0	0
Reissner-Nordstrøm Lösung	M	Q	0
Kerr Lösung	M	0	J
Kerr-Newman Lösung	M	Q	J

Die Fluchtgeschwindigkeit v_{fl} eines Körpers aus dem Gravitationsfeld einer Quelle der Masse M_Q ist durch Gleichung 2.1 gegeben. r entspricht dabei dem Abstand vom Zentrum der Quelle, G der Gravitationskonstanten.

$$v_{fl} = \sqrt{\frac{2GM_Q}{r}} \tag{2.1}$$

Ist die Masse nun so stark komprimiert, dass die Fluchtgeschwindigkeit v_{fl} bei einem Abstand r der Lichtgeschwindigkeit c entspricht, so erhält man durch Umstellen nach r die Gleichung 2.2.

$$r_S = \frac{2GM_Q}{c^2} \tag{2.2}$$

Dies ist der sogenannte Schwarzschildradius. Er ergibt sich ebenfalls durch das Lösen der Einsteinschen Feldgleichung. Näheres dazu ist in der Vorlesung von G. 't Hooft ('t Hooft, 2009, Kapitel 6, S. 11) zu finden. Für ein elektrisch neutrales, statisches Schwarzes Loch ist der Schwarzschildradius gleichbedeutend mit dem Ereignishorizont. Dies ist die Fläche, auf der die Fluchtgeschwindigkeit v_{fl} der Lichtgeschwindigkeit c entspricht. Sobald von der Schwarzschild-Lösung abgewichen wird, das Schwarze Loch also Ladung oder einen Drehimpuls besitzt, liegen Schwarzschildradius und Ereignishorizont nicht mehr aufeinander. Auf diese Lösungen soll in dieser Arbeit nicht weiter eingegangen werden. Sie werden jedoch beispielsweise in 't Hoofts Vorlesung ('t Hooft, 2009) näher erläutert.

2.1.3 Primordiale Schwarze Löcher

Die Existenz primordialer Schwarzer Löcher ist nicht erwiesen, der Theorie nach jedoch möglich. Für ihre Entstehung gibt es unterschiedliche Erklärungsansätze. Die Kompression von Materie zu einem Schwarzen Loch würde sehr viel Energie erfordern. Da die Chandrasekhar-Grenze ihre Bildung durch einen Gravitationkollaps ausschließt, kann daraus gefolgert werden, dass sich solche kleinen Löcher nur im primordialen Universum gebildet haben können (R. Bousso, 1996, Kapitel 1, S.2).
Ein möglicher Ansatz berücksichtigt Dichteschwankungen. In Regionen mit ausreichend hoher Dichte könnte sich durch den Kollaps der Materie ein kleines schwarzes Loch gebildet haben (G. D. Kribs, 1999, S. 2).
Hawking schlägt in seiner Veröffentlichung (R. Bousso, 1996, Kapitel 1.3, S.3f) von 1996 vor, dass es durch Fluktuationen im frühen Universum, die eine Änderung der Topologie der Raumzeit hervorrufen könnten, zur Paarbildung Schwarzer Löcher gekommen sein könnte. Durch den inflationären Zustand wären die Schwarzen Löcher auseinander gezogen worden, sodass die Rekombination verhindert worden wäre.
Im Jahr 1974 stellte Hawking fest, dass Schwarze Löcher thermische Strahlung mit dem Spektrum eines Schwarzkörperstrahlers emittieren. Diese sogenannte *Hawking-Strahlung* begründete er durch Vakuumfluktuationen nahe des Ereignishorizonts. Wird ein virtuelles Teilchenpaar aus Photonen knapp außerhalb des Ereignishorizonts erzeugt, so tunnelt das virtuelle Teilchen negativer Energie in das Schwarze Loch. Das Teilchen positiver Energie entschwindet. Dadurch verliert das Schwarze Loch an Masse. Dieser Ansatz wird unter anderem in B. Schutz' Buch (Schutz, 2004, S.304) erläutert.
In eben diesem Buch (Schutz, 2004, S. 304f) wird über diesen Ansatz gezeigt, dass die Luminosität eines Schwarzen Loches proportional zu M^{-2} ist. Je leichter das Schwarze Loch, desto stärker evaporiert es auf Grund von Hawking-Strahlung. Für ein ungeladenes, nicht rotierendes Schwarzes Loch kann die Massenabnahme mit Gleichung 2.3 beschrieben werden. α entspricht dabei einer masseabhängigen Größe.

$$\frac{dM}{dt} = -\frac{\alpha}{M^2} \tag{2.3}$$

Durch Integration und den Ansatz $M(t_{PBH}) = 0$ kann die Lebenszeit eines primordialen Schwarzen Loches bestimmt werden. Sie ist in Gleichung 2.4 dargestellt. M_i entspricht der ursprünglichen Masse des Schwarzen Loches.

$$t_{PBH} = \frac{M_i^3}{3\alpha} \tag{2.4}$$

Die Lebensdauer eines Schwarzen Loches mit der Masse der Sonne würde in etwa 10^{66} Jahre betragen.

Auf diese Weise wird in der Veröffentlichung von Overduin und Wesson (J.M. Overduin, 2004, Kapitel 9.2, S. 146ff) die Masse der primordialen Schwarzen Löcher berechnet, die in der heutigen Zeit, d.h. nach etwa der Hubblezeit[4], mit einer Explosion enden würden. Diese Masse ergibt sich zu $M_{PBH} = (5{,}7 \pm 1{,}4) \cdot 10^{-14}\,g$. Desweiteren wird über diesen Ansatz die Anzahldichte primordialer Schwarzer Löcher abgeschätzt. Es folgt eine Massendichte von $\rho_{PBH} \approx 5{,}74 \cdot 10^{18}\,\frac{g}{pc^3}$.

2.2 Poisson-Verteilung

Zählexperimenten können verschiedene Wahrscheinlichkeitsverteilungen zugeordnet werden. Handelt es sich um ein Bernoulli-Experiment, so nennt sich die zugehörige Wahrscheinlichkeitsverteilung Binomialverteilung. Ein Bernoulli-Experiment ist dadurch charakterisiert, dass nur das Eintreten („Erfolg“) bzw. das Nichteintreten („Misserfolg“) eines Ereignisses von Bedeutung ist. Die Erfolgswahrscheinlichkeit ist dabei konstant. Die Binomialverteilung lässt sich wie folgt darstellen:

$$P(n) = \binom{N}{n} \cdot p^n \cdot (1-p)^n = \frac{N!}{(N-n)! \cdot n!} \cdot p^n \cdot (1-p)^{N-n}$$

Dabei entspricht N dem Stichprobenumfang, n der Anzahl der „Erfolge“ und p der Erfolgswahrscheinlichkeit.

Für einen großen Stichprobenumfang N und kleine Erfolgswahrscheinlichkeiten p geht diese Verteilung in die sogenannte Poisson-Verteilung über.
Mit dem Erwartungswert $\lambda = N \cdot p = \text{const.}$ und für $N \gg n$, $p \ll 1$ folgt:

$$\begin{aligned}\lim_{N\to\infty} P(n) &= \lim_{N\to\infty} \frac{N!}{(N-n)! \cdot n!} \cdot \left(\frac{\lambda}{N}\right)^n \cdot \left(1 - \frac{\lambda}{N}\right)^{N-n} \\ &= \lim_{N\to\infty} \frac{\lambda^n}{n!} \cdot \underbrace{\left(\frac{N!}{(N-n)! \cdot N^n}\right)}_{\to 1} \cdot \underbrace{\left(1 - \frac{\lambda}{N}\right)^N}_{\to \exp^{-\lambda}} \cdot \underbrace{\left(1 - \frac{\lambda}{N}\right)^{-n}}_{\to 1}\end{aligned}$$

$$P(n,\lambda) = \frac{\lambda^n \cdot e^{-\lambda}}{n!} \tag{2.5}$$

Für Erwartungswerte $\lambda \ll 1$ ist die Wahrscheinlichkeit n=1 Ereignis zu beobachten:

$$P(1,\lambda \ll 1) = \frac{\lambda \cdot e^{-\lambda}}{1} \overset{\lambda \ll 1}{\approx} \lambda \cdot (1 - \lambda + \mathcal{O}(\lambda^2))$$

$$P(1,\lambda \ll 1) \approx \lambda \tag{2.6}$$

Die Wahrscheinlichkeit kann also durch den Erwartungswert selbst genähert werden.

[4] $T = 1/H_0$; mit der Hubble-Konstanten $H_0 = 0{,}102 \cdot h_0 \cdot 10^{-9}\,a$, h_0 ist nicht genau bestimmt und liegt zwischen $0{,}6 \leq h_0 \leq 0{,}9$

2.3 Das Fermi Gamma-ray Space Telescope

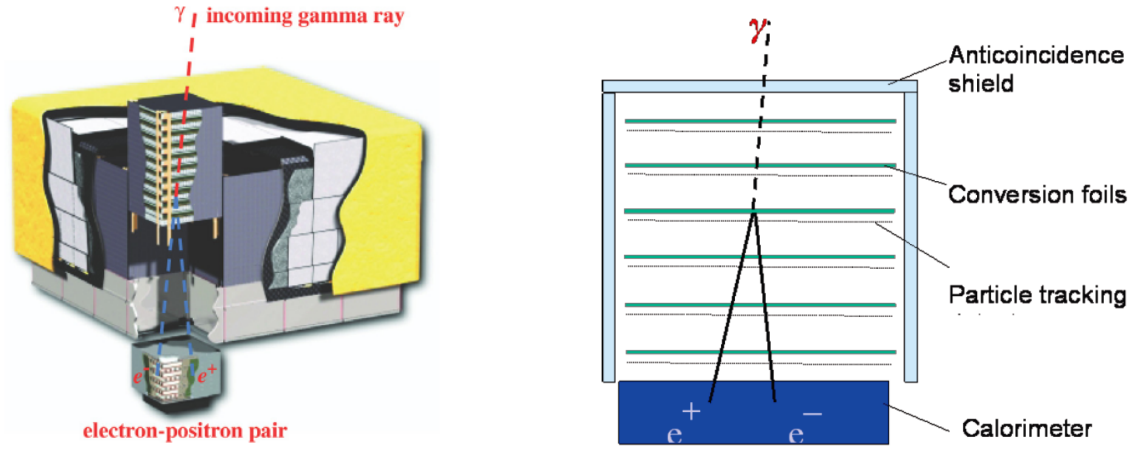
Bis heute gibt es viele bisher ungeklärte Fragen zum hochenergetischen Universum. Dazu gehören zum Beispiel die Suche nach neuen physikalischen Gesetzen im hochenergetischen Bereich, die Suche nach dunkler Materie und die Frage nach der Entstehung von Gamma-ray Bursts. Das 1991 gestartete *Energetic Gamma Ray Experiment Telescope* (EGRET) vollführte zum ersten Mal eine Vermessung des gesamten Himmels im Energiebereich von 30 MeV bis 10 GeV und entdeckte dabei viele bisher unidentifizierte Gammastrahlenquellen. Das *Fermi Gamma-ray Space Telescope* (Fermi) wurde in Hinsicht auf EGRET in vielen Gesichtspunkten, wie zum Beispiel der Sensitivität, der Totzeit oder dem Sichtbereich (engl.: Field of View, FOV) verbessert. Mit den Daten des Fermi-Teleskops sollen von EGRET entdeckte Quellen identifiziert und offene Fragen in Bezug auf hochenergetische Ereignisse im Universum, wie beispielsweise die Ursache für Gamma-ray Bursts oder die Frage nach Kandidaten für Dunkle Materie geklärt werden (Atwood et al., 2009).
Das Fermi-Teleskop startete am 11. Juni 2008 von Cape Canaveral aus. Es deckt einen Energiebereich von etwa 10 keV bis zu 300 GeV ab und ist in das *Large Area Telescope* (LAT) (siehe Abb. 2.3(a)) und den *Gamma-ray Burst Monitor* (GBM) unterteilt. Die kurze Totzeit des LAT ist besonders gut geeignet, um auch transiente Ereignisse zu beobachten (Atwood et al., 2009).
Da die Datenauswertung in dieser Arbeit ausschließlich auf den vom LAT aufgenommenen Daten beruht, wird auf den GBM im Folgenden nicht näher eingegangen.

2.3.1 Das Large Area Telescope

Die primäre Aufgabe des LAT ist es, Photonen aus dem Energiebereich von etwa 20 MeV bis zu 300 GeV zu detektieren. Außerdem kann die Energie und die Trajektorie des Photons mittels Algorithmen rekonstruiert werden. Das LAT ist in je 16 Spurdetektoren und 16 Kalorimeterblöcke unterteilt. Photonen werden im Spurdetektor über den Prozess der Paarerzeugung nachgewiesen. Geladene Hintergrundsignale [5] können über einen Antikoinzidenz-Detektor (ACD: anticoincidence detector) ausgeschlossen werden (siehe Abb. 2.3(a)). Das LAT hat einen besonders großen Sichtbereich[6]. Dieser beträgt etwa 2.4 sr und somit ist das LAT in der Lage ca. 20% des Himmels auf einmal zu beobachten (Atwood et al., 2009). Ein Umlauf um die Erde dauert in etwa 96 min. Nach etwa 3 Stunden hat das Teleskop seine ursprüngliche Position erneut erreicht und jeden Bereich des Himmels dabei gleichmäßig für etwa 30 min beobachtet (Bouvier, 2010, Kapitel 4.2).

[5] Geladene Hintergrundereignisse können beispielsweise durch die kosmische Hintergrundstrahlung erzeugt werden.

[6] $FoV = \int A_{eff}(\theta,\phi)d\Omega / A_{eff}(0,0)$ bei 1 GeV, mit A_{eff} als effektive Fläche des LAT abzüglich aller Schnitte auf den Hintergrund



(a) (b)

Abbildung 2.3: (a) Schematischer Aufbau des Large Area Telescope mit den Tracker-Modulen (blau), den Kalorimeterblöcken (grau) und dem Antikoinzidenz-Detektor (gelb). Die Größe des Teleskops beträgt $1.8\,\mathrm{m} \times 1.8\,\mathrm{m} \times 0.72\,\mathrm{m}$ (Atwood et al., 2009). (b) Schematischer Aufbau eines Tracker- sowie Kalorimetermoduls. Ein eintreffendes Photon konvertiert an einer Wolframschicht in ein e^+e^--Paar, sodass deren Positionen im SSD (silicon-strip detectors) bestimmt werden können (Bouvier, 2010).

Die Totzeit des LAT beträgt etwa 26.5 µs, sodass auch transiente Ereignisse gut beobachtet werden können. Desweiteren hat das LAT eine große effektive Fläche von in der Regel über $8000\,\mathrm{cm}^2$ [7] (Atwood et al., 2009). Einige Daten über das LAT sind in Abbildung 2.4 zusammengefasst.

2.3.1.1 Konversions-Tracker

Der Spurdetektor besteht aus 4×4 einzelnen Modulen, die jeweils im Abstand von 18 mm zueinander angeordnet sind. Jedes dieser Module besteht aus 18 Detektionsebenen, die wiederum in zwei Lagen (in x- und y-Richtung) von Siliziumdetektoren (SSDs: silicon-strip detectors) unterteilt sind. Die oberen 16 dieser Detektorschichten werden von Wolfram (Z=74), als Konversionsmaterial abgeschirmt, um eine Konversion eines eindringenden Photons in ein e^+e^--Paar hervorzurufen (siehe Abb. 2.3(b)). Wolfram wurde u.a. auf Grund seiner hohen Ordnungszahl gewählt, um die Wahrscheinlichkeit der Konversion zu erhöhen. Ein eintreffendes Photon konvertiert somit entweder an einer der Wolfram-Schichten in ein e^+e^--Paar, dessen x- und y-Positionen anschließend über die SSDs aufgezeichnet werden können, oder es kommt in einem der Detektoren zu Compton-Streuung.

[7] Effekte wie Reflexion, Vignettierung, Effizienz des Detektors, Position des Detektorrs und viele andere sorgen bei einem Teleskop dafür, dass das aufgenommene Signal abgeschwächt ist. Dieses abgeschwächte Signal kann dazu verwendet werden, auszurechnen, welche Fläche ein perfektes Teleskop haben würde, bei dem alle eingangs genannten Effekte nicht auftreten würden. Die theoretische Fläche eines angenommen perfekten Teleskopes nennt man „effektive Fläche“.

Parameter	Value or Range
Energy range	20 MeV–300 GeV
Effective area at normal incidence[a]	9,500 cm^2
Energy resolution (equivalent Gaussian 1σ):	
100 MeV–1 GeV (on-axis)	9%–15%
1 GeV–10 GeV (on-axis)	8%–9%
10 GeV–300 GeV (on-axis)	8.5%–18%
>10 GeV (>60° incidence)	⩽6%
Single photon angular resolution (space angle)	
on-axis, 68% containment radius:	
>10 GeV	⩽0.°15
1 GeV	0.°6
100 MeV	3.°5
on-axis, 95% containment radius	$< 3 \times \theta_{68\%}$
off-axis containment radius at 55°	< 1.7× on-axis value
Field of View (FoV)	2.4 sr
Timing accuracy	$< 10\ \mu s$
Event readout time (dead time)	26.5 μs
GRB location accuracy onboard[b]	$< 10'$
GRB notification time to spacecraft[c]	<5 sec
Point source location determination[d]	$< 0.'5$
Point source sensitivity (>100 MeV)[e]	3×10^{-9} ph cm^{-2} s^{-1}

Abbildung 2.4: Übersicht über wichtige Daten des LAT (Atwood et al., 2009).

Mehrfachstreuung, die zu einer Ablenkung des Photons von seiner ursprünglichen Bahn führen würde, wird durch geringe Abstände zwischen Wolframschicht und SSDs minimiert (Atwood et al., 2009) (Bouvier, 2010, Kapitel 4.3, S. 50ff).

2.3.1.2 Kalorimeter

Das Kalorimeter besteht wie der Tracker aus 4×4 Modulen, sodass jedem Trackermodul ein Kalorimetermodul zugeordnet werden kann. Jedes Modul setzt sich aus je 8 Ebenen mit jeweils 12 CsI(TI)-Kristallen zusammen. Die Kristalle sind optisch isoliert und horizontal ausgerichtet. In 4 dieser Ebenen sind die Kristalle in x-Richtung und in 4 Ebenen in y-Richtung orientiert, um eine dreidimensionale Lokalisierung des elektromagnetischen Schauers zu ermöglichen. Die als Szintillatoren fungierenden Kristalle sind an Photodioden angeschlossen. Durch die Elektronen bzw. Positronen, die durch Paarproduktion im Tracker entstehen, wird im Kalorimeter ein elektromagnetischer Schauer ausgelöst. Sowohl die genaue Position des Schauers, als auch die deponierte Energie können im Kalorimeter gemessen werden (Bouvier, 2010, Kapitel 4.3.2). Sollte ein Photon mit so viel Energie in den Tracker eindringen, dass nicht die gesamte Energie des Schauers im Kalorimeter deponiert wird, gibt es Algorithmen, die die verlorene Energie rekonstruieren. Gleiches gilt für einen Schauer, der eine Lücke zwischen zwei Kalorimeterblöcken passiert. Eine ausführlichere Beschreibung dieser Algorithmen ist in der Veröffentlichung von Atwood et al. (Atwood et al., 2009) gegeben.

2.3.1.3 Antikoinzidenz-Detektor

Der Antikoinzidenz-Detektor umschließt sämtliche Trackermodule. Er ist in viele kleine Abschnitte unterteilt, die alle aus Szintillatormaterial bestehen. Dringt also ein geladenes Teilchen von außen in das Teleskop ein, so entsteht an der entsprechenden Stelle ein Signal. Auf diese Weise können Photonen von geladenen Teilchen unterschieden werden.
Ein wichtiges Merkmal ist die Segmentierung des ACD, welche auf den sogenannten *backsplash-Effekt* zurückzuführen ist. Bei der Schauerbildung im Kalorimeter werden die Sekundärteilchen isotrop abgestrahlt. Diese können zur Comptonstreuung im ACD und damit zu einem verfälschten Signal führen. Demnach wird ein Ereignis nur verworfen, wenn ein ACD-Segment entlang der Teilchenbahn triggert (Atwood et al., 2009).

Misst das LAT ein Ereignis, so übermittelt es die Positionen der Treffer, sogenannten *hits*, in Tracker, Kalorimeter und ACD an das zuständige Zetrum auf der Erde, das *Mission Operations Center* (MOC) genannt wird (Bouvier, 2010, Kapitel 4.3.4). Mehrere Algorithmen dienen nun dazu, das Ereignis zu rekonstruieren. Im Falle eines Photonenereignisses werden charakteristische Parameter, wie die Richtung des einfallenden Photons und dessen Energie in einer Datei, die als *photon file* bezeichnet wird, gespeichert. Informationen über Position und Ausrichtung des Teleskops hingegen werden in einem sogenannten *spacecraft file* festgehalten (Fermi Science Support Center, a).

2.4 Software

Zur Analyse und Aufbereitung der Daten wurde in dieser Arbeit auf verschiedene, frei erhältliche Softwarepakete zurückgegriffen. Zum besseren Verständnis der Funktionsweise werden diese Pakete im Folgenden kurz beschrieben.

2.4.1 Fermi Science Tools

Auf der Fermi-Internetseite (NASA, d) können die Dateien *spacecraft file* und *photon file* heruntergeladen werden. Alle dort zur Verfügung gestellten Daten befinden sich im FITS-Dateiformat. Nähere Informationen zu diesem Format sind auf der Internetseite „The FITS Support Office“ (HEASARC, 2011) des HEASARC zu finden.
Das *spacecraft file* enthält Informationen über Position und Ausrichtung des Teleskops in Relation zur Erde. Diese werden im 30 s Rhythmus angegeben. Desweiteren wird gespeichert, zu welchen Zeiten das Teleskop voll einsatzfähig war, ob das Teleskop der Südatlantischen Anomalie [8] (engl.: South Atlantic Anomaly) ausgesetzt war und wie gut die Qualität der

[8] Durch das Magnetfeld der Erde werden geladene Teilchen im sogenannten *Van-Allen-Gürtel* eingefangen. Vor der Küste Brasiliens über dem Südatlantik ist das Magnetfeld am schwächsten und die Teilchen nähern sich der Erdoberfläche. Damit ist auch das Fermi-Teleskop beim Durchqueren dieser Gegend einer erhöhten Strahlung ausgesetzt (F. Fürst et al., 2009). Etwa 15% der Zeit befindet sich Fermi in der Nähe der Südatlantikanomalie (Fermi Science Support Center, d).

Daten ist.

Das *photon file* enthält Informationen zu jedem detektierten Ereignis. Es sind zum Beispiel die Energie, die rekonstruierte Richtung, aus der das Photon stammt, und der Zenitwinkel, sowie die Zeit, zu der das Ereignis detektiert wurde in Relation zur Startzeit der Fermimission[9], gespeichert.

Alle Angaben, die in den Dateien *spacecraft file* und *photon file* gespeichert sind, sind auf der Internetseite der NASA (NASA, c) aufgelistet.

Für diese Arbeit wurde auf die Photonendaten in Form von wöchentlichen Dateien zurückgegriffen. Eine solche wöchentliche Datei enthält Angaben zu jedem Ereignis, das in der jeweiligen Woche gemessen wurde. Zur Auswertung der von Fermi gemessenen Ereignisdaten steht frei erhältliche Software der Fermi-Kollaboration (Fermi Science Support Center, c) zur Verfügung. Diese ermöglicht beispielsweise das Herausfiltern einzelner Koordinaten oder Energiebereiche aus den Rohdaten. Desweiteren kann die Qualität der Daten über die Auswahl einer sogenannten Ereignisklasse beeinflusst werden.

Ereignisklassen Entsprechend der Wahrscheinlichkeit ein Photonereignis zu sein, werden die von Fermi detektierten Ereignisse Ereignisklassen zugeordnet. Dazu bietet die aktuelle Version der LAT-Daten *pass 7* der NASA 4 verschiedene Klassifizierungen an (Fermi Science Support Center, 2012).

1. Transient
2. Source
3. Diffuse
4. Data Clean

Diese Ereignisklassen sind ineinander verschachtelt, sodass die *Diffuse*-Klasse alle *Data Clean*-Ereignisse, die *Source*-Klasse alle *Diffuse*-Ereignisse und die *Transient*-Klasse alle *Source*-Ereignisse enthält. Demnach ist die *Data Clean*-Klasse die strikteste Ereignisklasse von allen. Sie lässt kaum Hintergrundereignisse zu und berücksichtigt lediglich eine kleine effektive Fläche. Dies bedeutet allerdings auch, dass die Quantität der Ereignisse sinkt. Somit ist die *Data Clean*-Klasse besonders für die Untersuchung großer Flächen geeignet, da so auch genügend Ereignisse guter Qualität vorliegen.
Die *Transient*-Klasse hingegen toleriert zugunsten einer hohen Statistik viel Hintergrund. So können auch fälschlicherweise als Photon rekonstruierte Ereignisse von geladenen Hintergrundteilchen zugelassen werden. Diese Klasse wird für kurzzeitige Ereignisse mit vielen Photonen, wie beispielsweise Gamma-ray Bursts empfohlen. Welche Ereignisklasse für welche Art von

[9] mission elapsed time (MET): 01.01.2001, 0h:0m:0s UTC. (NASA, d)

Untersuchungen von der NASA vorgeschlagen werden, kann sowohl für *pass6*-Daten (Fermi Science Support Center, a) bzw. für *pass7*-Daten in (Fermi Science Support Center, 2012) auf der Internetseite der NASA nachgelesen werden. Eine genaue Beschreibung der einzelnen Klassen ist in der Veröffentlichung von Atwood et al. (Atwood et al., 2009) zu finden.

Für diese Arbeit wurden die zwei Programme *gtselect* und *gtmktime* genutzt, um die Daten zu filtern. Die Funktionsweise beider Programme soll im Folgenden kurz erläutert werden.

gtselect *gtselect* ist ein Programm, welches benutzerdefinierte Schnitte auf vorgegebenen Ereignisdaten macht. Es sucht aus den Rohdaten die Ereignisse heraus, die vorgegebenen Schnitten genügen. So werden Ereignisse im Zeitintervall von T_{min} bis T_{max} in einem Energiebereich von E_{min} bis E_{max} ausgewählt. Auch eine Einschränkung auf einen Bereich am Himmel, in Form von Ra und Dec (J2000) Koordinaten und einen Radius, ist möglich. Ein Zentiwinkelschnitt ist wählbar, um beispielsweise den Einfluss der Erde zu minimieren bzw. bestenfalls zu eliminieren. Als Einfluss der Erde kommen Photonen infrage, die von dem LAT gemessen und als Hintergrundereignisse interpretiert werden würden.
Zusätzlich können über versteckte Parameter noch eine Reihe von zusätzlichen Schnitten gemacht werden. So kann beispielsweise die Ereignisklasse eingeschränkt werden (NASA, a). Eine Übersicht über alle gemachten Schnitte ist in Kapitel 3.2 in Tabelle 3.1 zu finden.

gtmktime Neben *gtselect* wurde das Tool *gtmktime* benutzt, welches ebenfalls Photonenereignisse filtert. *gtmktime* liest ein *spacecraft file* ein. Mit Hilfe dieser Datei können „gute“ Zeitintervalle, sogenannte *Good Time Intervals* (GTIs) ermittelt werden, während derer das Teleskop voll einsatzfähig war und die gesammelten Daten als vertrauenswürdig angesehen werden können. Die Rohdaten aus dem *photon file* werden nach GTIs durchsucht. Nur Ereignisse, die innerhalb dieser „guten“ Zeitintervalle detektiert wurden, werden in die Ausgabedatei übernommen. So werden beispielsweise Zeiten, in denen das Teleskop der Südatlantischen Anomalie ausgesetzt war, herausgefiltert (NASA, b). Zusätzlich können noch weitere Auswahlkriterien hinzugewählt werden. So wurde für die Datenselektion in dieser Arbeit ein Filter genutzt, der nur Ereignisse mit einer hohen Wahrscheinlichkeit, ein Photonenereignis zu sein, zulässt. Durch die Wahl dieses zusätzlichen Kriteriums wird die Quantität der Daten reduziert.

2.4.2 Healpix

Bei der Analyse der Daten des LAT ist es häufig sinnvoll, diese in einer Himmelskarte zu veranschaulichen. Dadurch können Unterschiede zwischen verschiedenen Bereichen des Himmels ausgemacht und verdeutlicht werden. In dieser Arbeit wird das Programm HEALPix verwendet. Dieser Name steht für *Hierarchical Equal Area isoLatitude Pixelization*. Es ist ein Programm zur Unterteilung einer gekrümmten Oberfläche in eine benutzerdefinierte Anzahl von Pixeln, die alle einen gleich großen Anteil der Oberfläche einnehmen. Diese Pixel können als zweidimensionale Karte dargestellt werden. Dazu wird die sogenannte Mollweideprojektion[10] verwendet, die eine flächengetreue Darstellung der Pixel bietet (Górski).

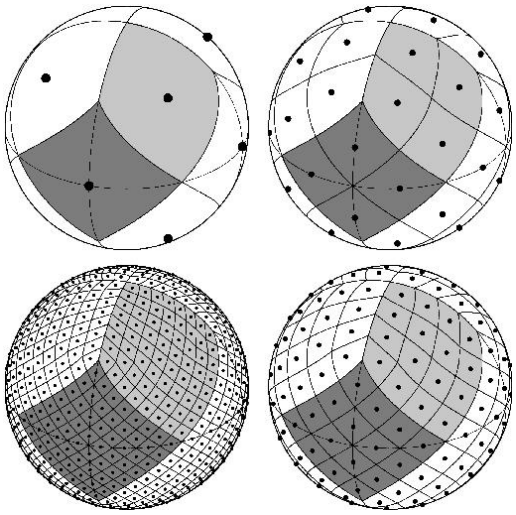


Abbildung 2.5: Unterteilung einer Kugeloberfläche in HEALPix-Pixel, oben links: $N_{side} = 1$, oben rechts: $N_{side} = 2$, unten links: $N_{side} = 8$ und unten rechts: $N_{side} = 4$; in verschiedenen Graustufen jeweils die Fläche eines der ursprünglichen Pixel bei $N_{side} = 1$ (Górski).

Die Anzahl der Pixel, in die die Oberfläche unterteilt werden kann, ist nicht beliebig wählbar. Die Pixelanzahl ergibt sich aus $12 \cdot \mathrm{N}_{\mathrm{side}}^2$, wobei $\mathrm{N}_{\mathrm{side}} = 2^n$ für $n \in \mathbb{N}$ entspricht. Die Mittelpunkte der Pixel liegen auf $(4 \cdot \mathrm{N}_{\mathrm{side}} - 1)$ Linien konstanter Breite. Damit kann eine Oberfläche minimal in 12 Pixel ($\mathrm{N}_{\mathrm{side}} = 1$), deren Mittelpunkte auf 3 Linien angeordnet sind, unterteilt werden. Die nächstgrößere Anzahl an Pixeln ist 48 ($\mathrm{N}_{\mathrm{side}} = 2$) mit 7 Linien gleicher Breite. Dabei wird jedes der 12 ursprünglichen Pixel in vier weitere unterteilt. Dies ist in Abbildung 2.5 dargestellt. Weitere Informationen und Hintergründe zu der Aufteilung einer Oberfläche in HEALPix-Pixel sind in einem Artikel von Górski et al. (Górski et al., 2005) gegeben. Bei der Wahl des $\mathrm{N}_{\mathrm{side}}$ ist darauf zu achten, dass dieses die Größe der Pixel beeinflusst. Je größer das $\mathrm{N}_{\mathrm{side}}$, desto kleiner die Fläche eines einzelnen Pixel. Ist ein Pixel besonders klein, so können

[10] Himmelskarte wird elliptisch dargestellt. Längengrade liegen auf Ellipsen, Breitengrade auf Geraden.

Ereignisse örtlich besser eingegrenzt und einer möglichen Quelle zugeordnet werden.

Sei N die Anzahl an Pixeln, so beträgt der Raumwinkel eines Pixel:

$$\Omega = \frac{4\pi}{N} \tag{2.7}$$

Über den halben Öffnungswinkel ϑ kann der Raumwinkel desweiteren angegeben werden durch:

$$\Omega = 2\pi \cdot (1 - \cos\vartheta) \approx \pi \cdot \vartheta^2 \tag{2.8}$$

$$\vartheta \approx \sqrt{\frac{\Omega}{\pi}} = \sqrt{\frac{4}{N}} \tag{2.9}$$

Im Gradmaß ergibt sich entsprechend:

$$\vartheta \approx \sqrt{\frac{4}{N}} \cdot \frac{180}{\pi} \tag{2.10}$$

Damit erhält man einen Quadratwinkel eines Pixels von:

$$\Theta = \pi \cdot \vartheta^2 \approx \frac{4}{N} \cdot \frac{180^2}{\pi} \tag{2.11}$$

Die Pixel werden durchnummeriert, sodass jedes einzelne Pixel anhand seiner Nummer identifiziert werden kann. Dabei kann für die Nummerierung zwischen den zwei Vorgehensweisen *ring* und *nested* unterschieden werden. Bei der *ring*-Variante, die ihrer Einfachheit halber in dieser Arbeit verwendet wurde, werden die Pixel aufsteigend von Nord nach Süd und entlängs der Breitengrade nummeriert (siehe Abb. 2.6). Jedem Koordinatenpaar (θ,ϕ) kann somit eine Pixelnummer zugeordnet werden. Bei der *nested*-Variante hingegen werden die Pixel entspre-

chend den 12 Basis-Pixeln ($N_{side} = 1$) sortiert. Genaueres dazu kann in der Anleitung zu HEALPIx (Górski, 2010) nachgelesen werden.

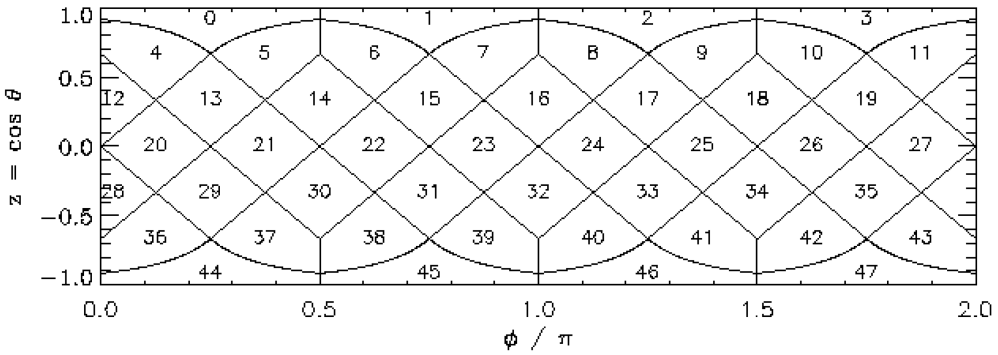


Abbildung 2.6: Schema der Pixelaufteilung einer zylindrischen Oberfläche bei $N_{side} = 2$ in der *ring*-Variante (Górski, 2010).

Wie bereits erwähnt, können mit HEALPix Himmelskarten in der Mollweideprojektion dargestellt werden. Diese Karte ist in HEALPix-Pixel unterteilt. In jedes Pixel kann eine beliebige Größe eingetragen werden. Im Verlauf dieser Arbeit werden mehrere solcher Karten genutzt, um verschiedene Größen darzustellen. Den Mittelpunkt einer solchen Himmelskarte bildet das Galaktische Zentrum.
Die Koordinaten sind in galaktischen Koordinaten angegeben. Die elliptisch dargestellten Längengrade laufen von 90° am oberen Rand der Karte bis -90° am unteren Rand, sodass im Zentrum 0° sind. Breitengrade verlaufen von 0° im Zentrum bis 179° am linken Rand und von 180° am rechten Rand bis 360° im Zentrum. Das Koordinatennetz ist im Abstand von 10° eingezeichnet. Verdeutlicht wird dies auch in Abbildung 2.7.

Das HEALPix-Paket wird für verschiedene Programmiersprachen, wie z.B. Python oder C, angeboten. Es enthält viele verschiedene Unterprogramme, beispielsweise ein Programm zur Umwandlung einer Koordinate in die entsprechende Pixelnummer. Eine Beschreibung sämtlicher Unterprogramme für die Programmiersprache C ist in (Hivon et al., 2010) gegeben. Zur Anwendung von *HEALPix* in Python ist eine eigenständige Version, das sogenannte *healpy* entwickelt worden, welches in dieser Arbeit zur Anwendung kommt.

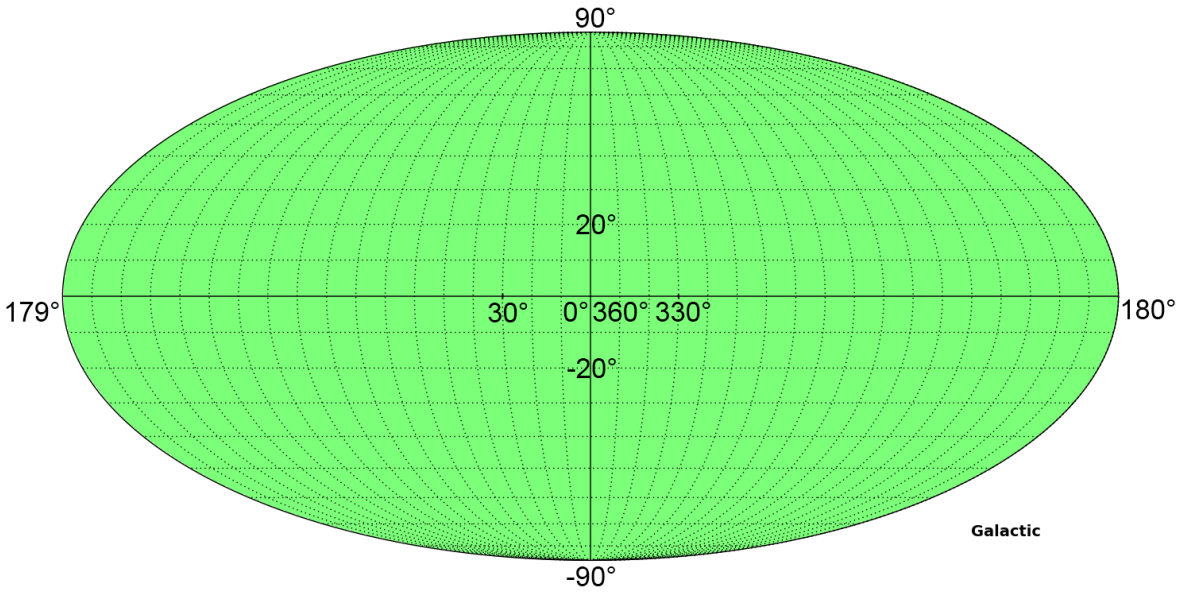


Abbildung 2.7: Himmelskarte in galaktischen Koordinaten zur Veranschaulichung der Mollweideprojektion.

Kapitel 3

Datensatz und -analyse

Ziel dieser Arbeit ist es, den Himmel in *HEALPix*-Pixel zu unterteilen und anschließend jedes Pixel auf das Auftreten eines transienten Ereignisses in der bisherigen Fermi-Laufzeit zu untersuchen. Zu diesem Zweck wurden die Daten aufbereitet und mehrere Programme entwickelt, die für einen Großteil der Analyse verantwortlich sind.

3.1 Methode

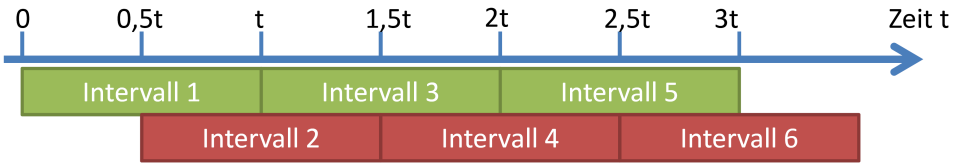


Abbildung 3.1: Schema der Unterteilung in 6 sich zur Hälfte überlappende Zeitintervalle mit der Dauer t

Auf der Suche nach transienten Ereignissen wird in dieser Arbeit der Fokus auf besonders kurzzeitige Ereignisse gelegt. Dazu wird der gesamte untersuchte Zeitraum in kleinere Zeitintervalle unterteilt. Benachbarte Zeitintervalle sollen sich gegenseitig überlappen (siehe Abb. 3.1), damit ein transientes Ereignis nicht zerteilt und möglicherweise nicht vollständig erkannt wird. Da ein Augenmerk auf kurzzeitige Ereignisse gelegt wird, werden Zeitintervalle von 1 s Dauer gewählt, die sich jeweils um 0.5 s überlappen. Somit können transiente Ereignisse, die innerhalb kürzester Zeit stattfinden, vollständig in einem Intervall erfasst werden. Im Laufe der

Arbeit stellte sich heraus, dass eine Unterteilung in Zeitintervalle von 1 s Dauer auf Grund der technischen Möglichkeiten und des zeitlich begrenzten Rahmens dieser Arbeit nicht möglich ist. Um die Rechendauer um einen Faktor 100 zu senken, wurde die Länge eines Zeitintervalls somit auf 100 s mit einem Versatz von 50 s erhöht.

Auf diese Weise kann jedem Zeitintervall eine Gesamtzahl an Ereignissen zugeordnet werden. Diese Daten werden histogrammiert, sodass nach Auffälligkeiten gesucht werden kann. Wird ein Kandidat für ein transientes Ereignis gefunden, so kann das zugehörige Pixel erneut mit einer Intervalllänge von 1 s untersucht werden. Mit Hilfe der Datenbank SIMBAD[1] kann in einem Radius von der halben Pixelausdehnung um die Koordinaten nach einer bekannten Gammaquelle gesucht werden (CDS).

3.2 Analyseparameter

Der in dieser Arbeit analysierte Datensatz wurde mit dem LAT aufgenommen. Zu Beginn der Datenauswertung lagen *photon files* der ersten 169 Wochen vor. Die Daten dieses gesamten Zeitraumes werden für die Untersuchung in dieser Arbeit herangezogen, um eine größtmögliche Datenmenge, in der ein transientes Ereignis beobachtet worden sein könnte, zu erhalten. Der Startzeitpunkt liegt bei der *mission start time* MST=239 557 417 s (31.07.2008) und der Endzeitpunkt der Daten bei T=341 366 402 s (27.10.2011), d.h. es wurden Daten aus ΔT=101 808 985 s verwendet. Dies entspricht etwa 3 Jahren und 3 Monaten.

Entsprechend der Zielsetzung, transiente Ereignisse im Energiebereich über 1 GeV zu finden, und die Einschränkungen, die durch die technischen Möglichkeiten des LAT gegeben sind, wurden die Daten mit dem Programm *gtselect* (siehe Kapitel 2.4.1) auf einen Energiebereich von 1 GeV bis 300 GeV reduziert.
Außerdem wurden die Daten auf die Ereignisklasse *Source* reduziert. Die *Transient*-Klasse würde, wie auf der NASA-Seite (Fermi Science Support Center, 2012) erwähnt, nur den Anteil an niederenergetischen Ereignissen steigern, beide sind aber gut zur Detektion transienter Ereignisse geeignet. Mit der Wahl der *Source*-Klasse wurde ein Anteil an Statistik eingebüßt, jedoch Qualität der Daten gewonnen. Mit der Reduktion auf die *Source*-Klasse werden im Energiebereich von 1 GeV bis 300 GeV 6857031 Ereignisse im Zeitraum der ersten 169 Wochen

[1] SIMBAD ist eine Datenbank, die die von verschiedenen Teleskopen detektierten Quellen katalogisiert hat. Es werden verschiedene Suchoptionen angeboten: Es können beispielsweise Informationen zu bekannten Quellen abgerufen oder zu einem Koordinatenpaar alle in der Nähe befindlichen Quellen aufgelistet werden.

verzeichnet. In der Tat enthält die *Transient*-Ereignisklasse für den untersuchten Zeitraum nur 2936 Ereignisse mehr. Dies entspricht etwa 0,000428%.
Die Datenklassen *Diffuse* und *Data Clean* würden die Statistik zu sehr reduzieren, als dass sie für diese Arbeit in Frage kämen. Eine größtmögliche Statistik ist für diese Arbeit sinnvoll, da eine Untersuchung einzelner Pixel stattfindet. Diese nehmen einen geringen Anteil am Himmel ein und weisen dementsprechend nur wenige Photonen mit Energien größer als 1 GeV pro Zeitintervall auf. Ein kurzzeitiges Ereignis kann in der Analyse von Hintergrundereignissen unterschieden werden.
In dieser Arbeit wurde ein Zenitwinkelschnitt bei 105° [2] gewählt, um den Einfluss der Erde bei 113° zu eliminieren. Anderenfalls würde die Statistik verfälscht werden.
Wie in Kapitel 2.4.1 schon erwähnt, wurde keine Einschränkung auf einen Ausschnitt des Himmels vorgenommen, weil am gesamten Himmel ein transientes Ereignis zu erwarten ist.
Bei der Wahl der Pixelanzahl waren drei Kriterien zu beachten. Zum einen ist die in Kapitel 2.4.2 erwähnte Größe eines Pixels zu beachten. Durch ein großes N_{side} würde die Fläche eines Pixels sehr klein und die Lokalisierung einzelner Photonquellen genauer. Weiterhin muss beachtet werden, dass eine größere Anzahl an Pixeln auch eine größere Rechenleistung erfordert. Zusätzlich ist das Auflösungsvermögen des LAT zu beachten, welches bei Ereignissen mit Energien von mehr als 1 GeV bei 0.6° (siehe Abb. 2.4) liegt.
Für ein $N_{side} = 64$ folgt ein Winkel von $\Theta \approx 0{,}84°$ (siehe Abschnitt 2.4.2), welcher ein wenig größer als das Auflösungsvermögen des LAT im verwendeten Energiebereich ist. Unter Berücksichtigung der technischen Möglichkeiten jedoch wurde in dieser Arbeit ein $N_{side} = 32$ gewählt, was 12288 Pixeln und einem Winkel $\Theta \approx 3{,}36°$ pro Pixel entspricht. Außerdem wurde diese Pixelanzahl bereits in der Masterarbeit von Jagdev Bains (Bains, 2011) verwendet und dient somit einer besseren Vergleichbarkeit der Ergebnisse. Eine Übersicht über alle gemachten Schnitte ist in Tabelle 3.1 zu finden.

[2] Im *gtselect*-Tutorial (Fermi Science Support Center, b) wurde zu Beginn der Arbeit ein Zenitwinkelschnitt von 105° vorgeschlagen. Dieser Wert wurde mittlerweile auf 100° korrigiert.

Tabelle 3.1: Übersicht über die gemachten Schnitte auf den Datensatz

Parameter	Wert
minimale Energie	1 GeV
maximale Energie	300 GeV
Startzeit	239 557 417 s
Endzeit	341 366 402 s
Analysedauer	101 808 985 s
Zenitwikelschnitt	105°
N_{side}	32
Pixelanzahl	12288

Im Folgenden soll zu jedem der zur Analyse verwendeten Programme ein Überblick über die Methodik gegeben werden, sodass der Zweck des Programmes deutlich wird. Anschließend wird detaillierter auf die Umsetzung eingegangen und zuletzt werden Probleme, die bei der Entwicklung des Programms entstanden sind, aufgezeigt.

3.3 Zuordnung der Pixelnummern

Das von Jagdev Bains im Rahmen seiner Masterarbeit entwickelte und in der Programmiersprache C geschriebene Programm „ang2HEALPix-print.c“ (Bains, 2011) dient dazu, jedes Photonereignis einem HEALPix-Pixel zuzuordnen und eine Karte zu erstellen, auf der die Anzahl an Ereignissen pro Pixel eingetragen ist.

Umsetzung Das Programm „ang2HEALPix-print.c“ liest die mit *gtselect* und *gtmktime* gefilterten LAT-Daten ein, um die einzelnen Ereignisse einem Pixel zuzuordnen.
Zusätzlich verlangt das Programm die Eingabe eine Ausgabedatei FITS-Dateiformat, sowie das gewünschte N_{side} zur Berechnung der Pixelanzahl, in die der Himmel unterteilt werden soll. Es liest zunächst zu jedem verzeichneten Photonereignis die galaktischen Koordinaten l und b, sowie den Zeitpunkt, an dem das Ereignis gemessen wurde, in jeweils eine Liste ein.

Anschließend werden die Koordinaten so transformiert, dass sie von der HEALPix-Software verarbeitet werden können. D.h. die Koordinate b wird von einer Skala, die von -90° bis +90° reicht, auf eine Skala von 0° bis 180° transformiert. Beide Koordinaten werden ins Bogenmaß umgerechnet. Daraufhin wird mit Hilfe des Wertes N_{side} über die *HEALPix*-Funktion „ang2pix_ring“ jedem Koordinatenpaar eine Pixelnummer aus dem *ring*-Schema (siehe Abschnitt 2.4.2) zugeordnet.
Das C-Programm gibt zunächst die Anzahl aller Ereignisse an, die in der Beobachtungszeit detektiert wurden und zu den gemachten Schnitten passen, und die Anzahl der Pixel, in die der Himmel unterteilt wurde. Anschließend wird eine Datei erstellt, in der jede Zeile einem Ereignis entspricht. Es werden jeweils Pixelnummer und Zeit in Relation zur Startzeit der Fermimission, zu der das Ereignis stattfand, angegeben. Schlussendlich ordnet das Programm jedem Pixel die Gesamtzahl an Ereignissen zu, die in diesem Bereich des Himmels im gesamten untersuchten Zeitraum detektiert wurden. Diese Daten werden mit der HEALPix-Funktion „write_healpix_map“ als Karte, der sogenannten *HEALPix-map* in der Ausgabedatei abgespeichert. Diese *HEALPix-map* kann über den Befehl „mollview“ in der Mollweideprojektion graphisch dargestellt werden. Informationen zu der HEALPix-map (Hivon et al., 2010) sind auf der Internetseite der NASA bereitgestellt. Die Himmelskarte für den in dieser Arbeit verwendeten Datensatz ist in Abbildung 3.2 zu sehen. Eine Abbildung mit absoluten Ereigniszahlen ist im Anhang zu finden.

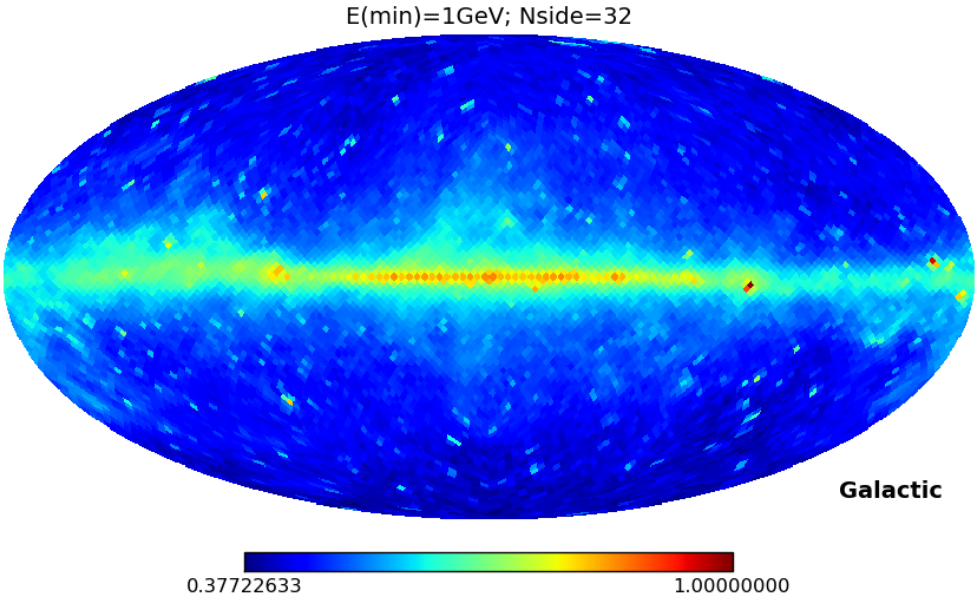


Abbildung 3.2: Himmelskarte mit den auf das Maximum normierten Ereignisanzahlen in jedem Pixel. Die Karte ist in 12288 Pixel unterteilt. Es werden nur Ereignisse aus dem Energiebereich von 1 GeV bis 300 GeV und aus den ersten 169 Wochen Datennahme berücksichtigt. Die Schnitte auf die Daten sind in Kapitel 3.2 beschrieben.

3.4 Ereignisse pro Pixel

Das in Python geschriebene Programm „Countsperpixel.py“ dient dem Zweck, für jedes Pixel eine Analyse einzelner Zeitintervalle (siehe Abschnitt 3.1) zu machen. Es werden innerhalb eines jeden 100 s langem Zeitintervalls die Anzahl der stattgefundenen Photonereignisse ermittelt. Zeitintervalle mit einem hohen Photonaufkommen könnten auf die Existenz eines transienten Ereignisses hindeuten.

Umsetzung Das Programm „Countsperpixel.py“ benötigt die zuvor mit dem in Kapitel 3.3 beschriebenen C-Programm erstellte Textdatei. Diese enthält sowohl die Parameterdaten, d.h. die Anzahl an Ereignissen und an Pixeln, als auch die Ereignisdaten, d.h. eine Auflistung der Pixelnummern und Zeiten zu den Ereignissen. Zur besseren Handhabung wird diese Datei in 2 separate Dateien aufgeteilt, sodass die Parameterdaten in einer Datei und die Ereignisdaten in einer zweiten Datei gespeichert sind. Diese beiden Dateien werden von dem Pythonprogramm als Eingabe benötigt.
Das Programm untersucht jedes Pixel einzeln, weil bei einer gleichzeitigen Analyse aller Pixel zu viel Arbeitsspeicher verbraucht würde. Dies wiederum hätte einen Abbruch des Programms zur Folge. Um mehrere Prozesse parallel zu starten, können ein Start- und ein Endpixel vorgegeben werden. Auf diese Weise kann eingegrenzt werden, welcher Teil des Himmels in diesem Arbeitsschritt untersucht werden soll.

Als erstes wird das angegebene Startpixel untersucht. Um im Folgenden Rechenzeit zu sparen, wird eine neue Liste angelegt, die nur die zu dem Pixel gehörigen Ereigniszeiten enthält. Im Anschluss werden alle $\frac{\Delta T-100}{50} \approx 2036178$ Zeitintervalle durchgegangen und nach Einträgen, die innerhalb dieses Zeitabschnitts liegen, durchsucht. Diese Anzahl an Zeitintervallen resultiert aus der Fermilaufzeit ΔT, der Intervalllänge von 100 s und dem Versatz der Zeitintervalle von 50 s (siehe Abschnitt 3.1). Ausgegeben wird eine Textdatei, welche für jedes Zeitintervall die dazugehörige Anzahl darin gefundener Ereignisse enthält. Dieses Vorgehen wird für die darauf folgenden Pixel analog durchgeführt, bis das Endpixel erreicht ist.

Probleme Dieses Programm wurde mehrfach umstrukturiert. Ursprünglich verarbeitete es alle Pixel und die gesamte Laufzeit auf einmal. Dafür reichte der zur Verfügung stehende Arbeitsspeicher jedoch nicht aus. Gelöst wurde dieses Problem zunächst durch die Aufteilung in einzelne Zeitabschnitte. So konnten über externe Parameter Start- und Endzeit des zu untersu-

chenden Zeitabschnitts vorgegeben werden. Um keine Speicherplatzfehler zu erhalten, wurden maximal 5 000 000 s in einem Durchlauf untersucht. Wie in der aktuellen Version des Programms wurden die Daten für jedes Pixel abgespeichert, allerdings nur für den untersuchten Zeitabschnitt. Somit resultierten am Ende der Auswertung für jedes Pixel über 100 einzelne Dateien[3]. Die Weiterverarbeitung dieser Daten gestaltete sich als äußerst problematisch. Die große Anzahl an erstellten Dateien, die alle eingelesen werden mussten, verlangsamte den Großrechner der Arbeitsgruppe so weit, dass er für weitere Mitglieder unbenutzbar wurde. Die in Kapitel 3.4 vorgestellte Methode erzeugt für jedes Pixel nur eine Datei. Das zuvor beschriebene Problem ist somit behoben. Außerdem wird ein Fehler auf Grund des Arbeitsspeichers vermieden, da jedes Pixel einzeln bearbeitet und die gespeicherten Größen für das nächste Pixel neu belegt und somit überspeichert werden.

3.5 Erstellen der Histogrammdaten

Aufgabe des Python-Programmes „CreateHist.py“ ist es, die aus dem Programm „Countsperpixel.py“ erhaltenen Daten zu histogrammieren. Dabei soll berücksichtigt werden, dass das Fermi-Teleskop zu jeder Zeit nur etwa ein Fünftel des Himmels betrachtet (siehe Abschnitt 2.3.1). Jedes andere, nicht beobachtete Pixel hat in diesem Zeitraum keinerlei Einträge. Da im GeV-Bereich jedoch nur wenige Ereignisse stattfinden[4], wird als Referenz der Energiebereich von 100 MeV bis 400 MeV hinzugezogen [5]. Sollte in mehreren aufeinander folgenden Zeitintervallen in beiden Energiebereichen für ein Pixel kein Photon aufgezeichnet werden, so kann angenommen werden, dass dieses Pixel in dieser Zeit nicht beobachtet wurde. In dieser Analyse wurde ein Block aus 33 aufeinanderfolgenden Intervallen, was einer Zeitdauer von 30 Minuten entspricht, gewählt, um die Beobachtung des Pixel von Fermi auszuschließen. Die Histogrammklassen sollen der Anzahl an Ereignissen entsprechen. Die Klassenhöhe der Anzahl an Zeitintervallen, in denen so viele Photonen detektiert wurden.

Umsetzung Das Programm „CreateHist.py “ verlangt nach der Eingabe einer Textdatei, in der die Histogrammdaten gespeichert werden sollen, sowie nach der Angabe eines Start- und eines Endpixels. Genau wie in dem Programm „Countsperpixel.py“ wird pixelweise vorgegangen.

[3] In der Regel wurden 1 000 000 s auf einmal untersucht. Bei einer Beobachtungszeit von 101 808 985 s resultieren etwas mehr als 100 Dateien pro Pixel. Insgesamt gibt es also über 1 200 000 Dateien.

[4] 6857031 Ereignisse in 169 Wochen und auf 12288 Pixel verteilt; dies entspricht durchschnittlich $5{,}5 \cdot 10^{-4}$ Ereignissen pro Zeitintervall und Pixel

[5] 153424266 Ereignisse in 169 Wochen und auf 12288 Pixel verteilt; dies entspricht durchschnittlich $1{,}2 \cdot 10^{-3}$ Ereignissen pro Zeitintervall und Pixel.

Zunächst werden für das Startpixel die Daten mit der Anzahl an Ereignissen pro Zeitintervall im GeV-Bereich und im MeV-Bereich aus den abgespeicherten Dateien eingelesen. Die Daten im GeV-Energiebereich werden auf 33 aufeinander folgende Zeitintervalle durchsucht, in denen kein Photon detektiert wurde. Weisen die selben Zeitintervalle im MeV-Bereich ebenfalls keine Ereignisse auf, so werden diese Zeitintervalle nicht mit in die Histogrammdaten aufgenommen. Zusätzlich werden auch die Randterme berücksichtigt. Sollten auf 33 hintereinander folgende Zeitintervalle ohne Ereignisse noch weitere (weniger als 33) Zeitintervalle folgen, in denen ebenfalls keine Ereignisse detektiert wurden, so werden auch diese Zeitintervalle nicht berücksichtigt.
Die Histogrammdaten werden in 20 Klassen unterteilt, deren Mittelpunkte bei 0 bis 19 liegen. Eine weitere Klasse umfasst alle Zeitintervalle mit mehr als 20 Ereignissen. Jede Klasse gibt an, wie viele Photonen in einem Zeitintervall detektiert wurden. Das selbe wird nun für die übrigen Pixel wiederholt. Abschließend werden die Histogrammdaten zu jedem Pixel in die zuvor angegebene Ausgabedatei eingetragen. Dabei stehen die Daten jedes Pixels in einer separaten Zeile. Diese enthalten, durch ein Leerzeichen getrennt, die Einträge der 20 Klassen, sowie in der letzten Spalte die Pixelnummer. Die Pixelnummern sind aufsteigend angeordnet.

Probleme Wie schon in Kapitel 3.4 erwähnt, wurden in diesem Programm ursprünglich für jedes Pixel viele Dateien eingelesen und anschließend aneinander gehängt. Dies sorgte zu einer deutlich längeren Laufzeit.
Das gewählte Referenzintervall wurde zunächst auf einen Energiebereich von 100 MeV bis 200 MeV festgelegt. Bei der weiteren Analyse fällt jedoch auf, dass in diesem Referenzintervall zu wenig Ereignisse[6] detektiert werden. Dies hat zur Folge, dass durch den Vergleich der Zeitintervalle in beiden Energiebereichen auch Zeitintervalle, in denen das Pixel von Fermi beobachtet wurde, nicht mit in die Histogrammdaten aufgenommen werden. Dies liegt daran, dass auch in der Zeit, in der das LAT auf ein Pixel gerichtet ist, in vielen Zeitintervallen keine Ereignisse gemessen werden. Dadurch werden die Ergebnisse in der weiteren Auswertung verfälscht. Eine Erweiterung des Energiebereichs auf eine noch größere Energiespanne ist in dieser Analyse nicht möglich gewesen, da der Arbeitsspeicher nicht ausreicht. Auf Möglichkeiten zur Verbesserung der Analyse wird in Kapitel 6 eingegangen.

[6] Im Energiebereich von 100 MeV bis 200 MeV wurden in der gesamten beobachteten Zeit 15 342 466 Ereignissen gemessen.

3.6 Auswahl interessanter Pixel

Die Hauptaufgabe des Python-Programmes „Interest_plot.py“ ist es, nach interessanten Pixeln zu suchen, in denen ein transientes Ereignis stattgefunden haben könnte. Dazu wird an die Histogrammdaten eine Exponentialfunktion

$$\lambda = \lambda_0 \cdot e^{-\alpha \cdot n_\gamma} \tag{3.1}$$

angepasst und nach Abweichungen von dieser gesucht. Die inverse Rate α soll in einer Himmelskarte dargestellt werden. Nahe der galaktischen Ebene werden mehr Photonereignisse erwartet als an den Polen. Somit wird eine zur galaktischen Ebene hin sinkende inverse Rate erwartet.

Umsetzung Das Python-Programm „Interest_plot.py“ verlangt als Eingabe die Histogrammdatei aus Kapitel 3.5, sowie die Angabe eines Start- und eines Endpixels. Es liest die Histogrammdaten klassenweise ein. An die Höhe der ersten zwei bzw. drei Klassen eines Pixels wird eine Exponentialfunktion angepasst. Zur einfacheren Handhabung der Exponentialfunktion werden die Klassenhöhen logarithmiert. Dadurch kann eine lineare Funktion (siehe Glg. 3.2) verwendet werden.

$$\ln(\lambda) = \ln(\lambda_0) - \alpha \cdot n_\gamma \tag{3.2}$$

Ein Beispiel dafür ist in Abbildung 3.3 zu finden.

Die inverse Rate α zu jedem Pixel erhält man also aus Gleichung 3.2.

$$\alpha = \frac{\ln(\frac{\lambda_0}{\lambda})}{n_\gamma} \tag{3.3}$$

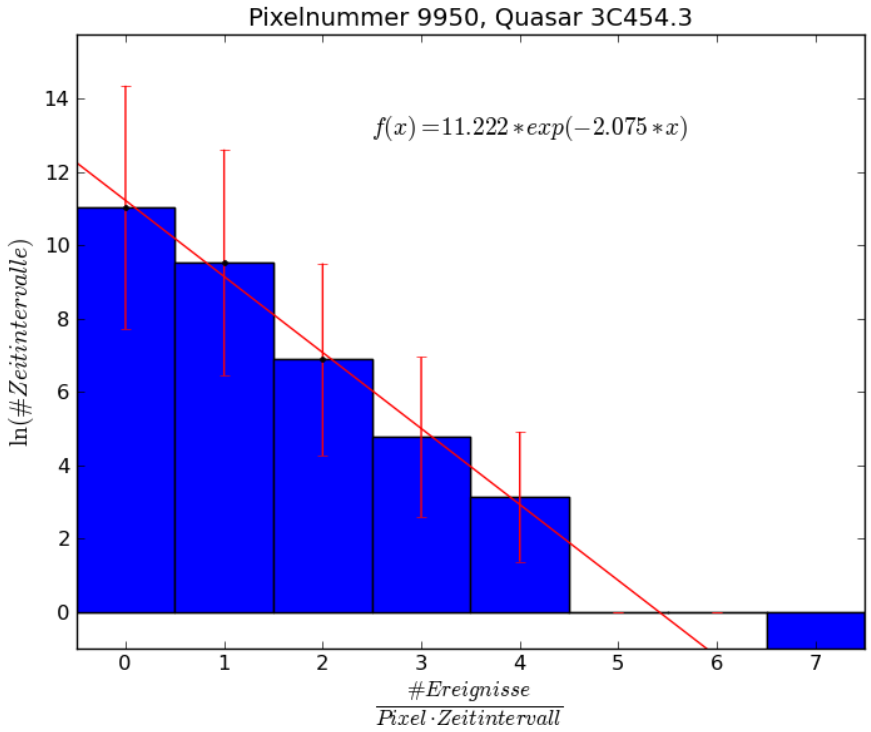


Abbildung 3.3: Angepasste Gerade an die Histogrammdaten des Pixels 9950. Dieses beinhaltet den Quasar 3C4543

Mit Hilfe der *numpy*-Funktion [7] *polyfit* kann an mehrere Datenpunkte eine Polynomfunktion angepasst werden, wobei der Grad der Funktion frei wählbar ist. Ein Zugriff auf die angepassten Parameter ist möglich. In der linearen Funktion $f(x) = p[0] \cdot x + p[1]$ entspricht $f(x) \hat{=} \ln(\lambda)$, $x \hat{=} n_\gamma$, der Parameter $p[0] \hat{=} -\alpha$ und der Parameter $p[1] \hat{=} \ln(\lambda_0)$. Somit ist für die inverse Rate α lediglich $-p[0]$ zu berechnen. Diese wird für sämtliche Pixel berechnet und anschließend mit Hilfe von *healpy* graphisch in einer Himmelskarte dargestellt.

Im Anschluss werden für interessante Pixel Histogramme erstellt. Ein Pixel wird als interessant gewertet, sobald eine logarithmierte Klassenhöhe einer Klasse mit mehr als 3 Ereignissen pro Zeitintervall über dem zugehörigen Wert der angepassten Geraden liegt. Für diese Pixel werden nun zum einen die Histogrammbalken mit logarithmierter Höhe und die angepasste Gerade in einer Graphik dargestellt.

Probleme Bei der Betrachtung der Histogrammdaten fällt auf, dass viele Pixel nur Einträge bei 0 bzw. 1 Ereignis pro Zeitintervall aufweisen. Eine Anpassung einer Geraden an die Logarithmen der ersten drei Klassen ist demnach nicht möglich, da der Logarithmus von 0 nicht

[7] *numpy* ist ein Modul, welches in der Programmiersprache Python genutzt werden kann.

existiert. Zur Berechnung der inversen Rate bietet es sich an, eine Unterscheidung zwischen den Pixeln vorzunehmen. Sollte das Pixel sowohl Einträge in den Klassen für 0 und 1 Ereignis pro Zeitintervall, als auch in der Klasse für 2 Zeitintervalle aufweisen, so kann eine Anpassung durch die ersten drei Punkte vorgenommen werden. Sind in der Klasse für 2 Ereignisse pro Zeitintervall keine Einträge, so wird die Anpassung nur für die ersten beiden Klassen durchgeführt. Das Darstellen des Logarithmus der Klassenhöhe 0 wurde durch das Addieren von 0,0001 ermöglicht. So wird die entsprechende Säule im Histogramm bis an die untere Grenze der y-Achse dargestellt, die anderen Werte in der Grafik jedoch nicht sichtbar verändert. Für die Anpassung der Geraden werden diese Werte nicht verwendet.
Wie in Kaipitel 3.5 erwähnt, werden zu wenige Intervalle mit Null Ereignissen in die Histogrammdaten aufgenommen. Dadurch wird die angepasste Funktion verfälscht. Der Betrag der Steigung α der angepassten Geraden wird zu klein.

3.7 Suche nach der Evaporation primordialer Schwarzer Löcher

In dem Programm „auswertung.py“ werden für die Suche nach der Evaporation primordialer Schwarzer Löcher relevante Parameter berechnet. Näheres dazu ist in Kapitel 5 zu lesen. Benötigt werden die Werte $\lambda(n_{\gamma,\max})$ und $\mathrm{n}_\gamma(\lambda = 10^{-5})$.

Umsetzung Genau wie das Programm „Interest_plot.py“ benötigt das Programm „auswertung.py“ die Histogrammdatei aus Kapitel 3.5 , die auch hier klassenweise eingelesen wird. Für jedes Pixel wird die maximale Anzahl von Ereignissen in einem Zeitintervall ermittelt. Diese werden benötigt, um den Wert $\lambda(\mathrm{n}_{\gamma,\max})$ zu berechnen. Dazu wird auf die Funktionsgleichung 3.1 zurückgegriffen und die maximale Anzahl $\mathrm{n}_{\gamma,\max}$ wird für n_γ eingesetzt. Das Ergebnis ist in Gleichung 3.4 zu sehen.

$$\lambda(n_{\gamma,max}) = \lambda_0 \cdot e^{-\alpha \cdot n_{\gamma,max}} \tag{3.4}$$

Zur Berechnung von $n_\gamma(\lambda = 10^{-5})$ wird Gleichung 3.1 nach n_γ aufgelöst und für λ wird der Wert 10^{-5} eingesetzt. Es folgt Gleichung 3.5.

$$n_\gamma(\lambda = 10^{-5}) = \frac{\ln(\frac{\lambda_0}{10^{-5}})}{\alpha} \tag{3.5}$$

Zuletzt wird der Mittelwert aus den Werten $n_\gamma(\lambda = 10^{-5})$ für alle Pixel berechnet.

Kapitel 4

Ergebnisse

In diesem Kapitel werden die bisherigen Ergebnisse der vorgestellten Analysemethode dargestellt. Auf Grund des begrenzten Zeitrahmens wurden nur 2000 Pixel (der insgesamt 12288 Pixel) untersucht. Als „interessant“ (siehe Abschnitt 3.6) klassifizierte Pixel werden vorgestellt. Anhand bekannter Gammastrahlen-Quellen wird gezeigt, dass die Analysemethode sensitiv auf transiente Ereignisse ist.

Eine Auswertung aller Pixel mit dem in Kapitel 3.7 beschriebenen Programm ergibt, dass pro Pixel im Durchschnitt maximal 1,2 Ereignisse in einem Zeitintervall stattfinden. Eine Wahrscheinlichkeit von $P = 10^{-5}$ zufällig eine Anzahl n Ereignisse in einem Zeitintervall zu detektieren, liegt durchschnittlich bei $n(\lambda = 10^{-5}) \approx 7{,}18$ Ereignissen.
Bei der Ermittlung dieser Werte wurde die Histogrammklasse 0 berücksichtigt. Dadurch ist die angepasste Funktion verfälscht und die Werte werden nicht korrekt ermittelt. Eine Anpassung ohne Null ist für viele Pixel jedoch nicht möglich, weil nur Zeitintervalle mit keinem bzw. einem detektierten Ereignis verzeichnet sind.

4.1 Hintergrund

Die Histogrammdaten vieler Pixel enthalten ausschließlich Hintergrundereignisse. In diesen Bereichen des Himmels hat Fermi keine aus einer Gamma-Quelle stammenden Photonen detektiert. Bei diesen Pixeln werden ausschließlich Zeitintervalle mit maximal 3 detektierten

Ereignissen verzeichnet. Die Verteilung der Ereignisse pro Zeitintervall eines solchen „Hintergrund“-Pixels ist in Form eines Histogramms in Abbildung 4.1 dargestellt. Jede Histogrammklasse entspricht einer gewissen Anzahl m von in einem Zeitintervall detektierten Ereignissen. Die Klassenhöhe gibt den natürlichen Logarithmus der Anzahl solcher Zeitintervalle mit m Ereignissen an. In rot ist eine Gerade dargstellt, die an die Höhen der ersten drei Klassen angepasst wurde. Die Fehlerbalken entsprechen der Wurzel der Klassenhöhe. Es wird deutlich, dass die Anzahl an Zeitintervallen mit m Ereignissen bei steigendem m monoton abnimmt. Es wurden nur Zeitintervalle mit m=0,1,2 Ereignissen gemessen.

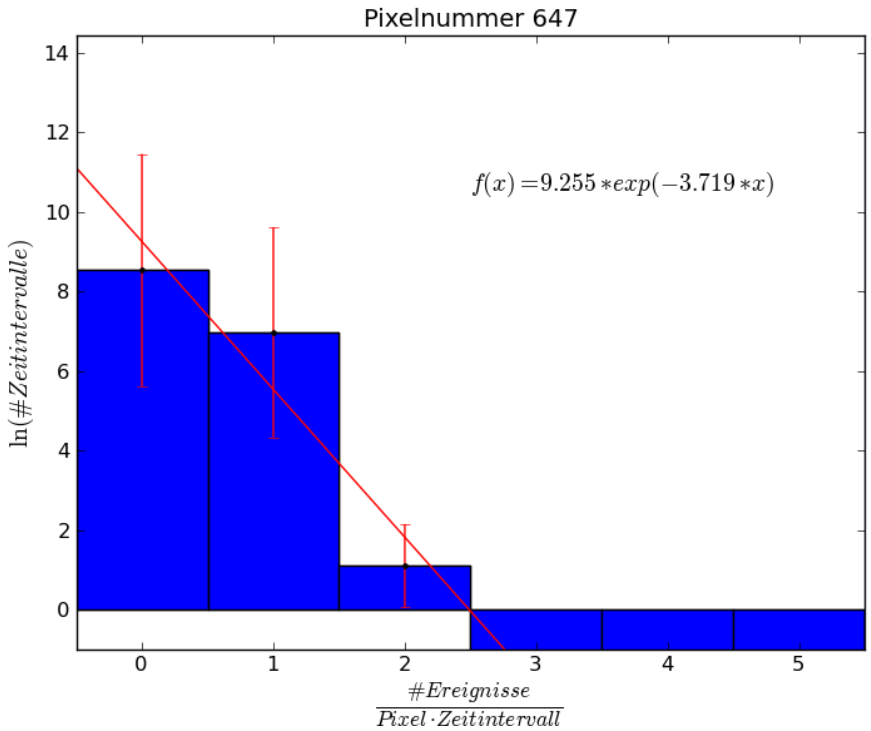


Abbildung 4.1: Histogrammdaten des Pixels 647. Es sind ausschließlich Hintergrundereignisse zu sehen.

4.2 Gamma-ray Bursts

In Kapitel 2.1.1 wurden vier von Fermi detektierte Gamma-ray Bursts vorgestellt, die mehrere Photonen im GeV-Bereich aufweisen. Ihre Positionen sind im selben Kapitel in Abbildung 2.2 eingetragen. Die genauen Koordinaten wurden durch eine Suche in der Datenbank SIMBAD ermittelt und mit Hilfe von *healpy* wurden die zugehörige Pixelnummern bestimmt.

Tabelle 4.1: Übersicht über 4 vom LAT bis Januar 2010 detektierte Gamma-ray Bursts mit Photonenergien $> 1\,\mathrm{GeV}$; deren Positionen in galaktischen Koordinaten angegeben sind. Die zugehörigen Pixelnummer werden aufgelistet.

GRB	L (J2000)	B (J2000)	Pixelnummer	# Ereignisse $> 1\,\mathrm{GeV}$
080916C	270.1480	-13.7598	7584	13
090510	024.5976	-55.0765	11094	$\sim$20
090902B	051.4952	+26.9110	3282	$\sim$30
090926A	314.6945	-48.9886	10870	$\sim$50

Da es sich auch bei Gamma-ray Bursts um transiente Ereignisse handelt, bei denen innerhalb eines Zeitintervalls viele Photonen ankommen, sollten sie mit dieser Analysemethode gefunden werden können. In Tabelle 4.1 sind 4 Gamma-ray Bursts genannt, die Photonen mit Energien $> 1\,\mathrm{GeV}$ aufweisen. Zu jedem Burst sind die mit der Datenbank SIMBAD ermittelten galaktischen Koordinaten und die entsprechende Pixelnummer angegeben.

Die Histogramme der Gamma-ray Bursts sind genau wie im Fall des Hintergrundpixels aufgebaut. Für die Gamma-ray Bursts GRB 090510, GRB 080916C und GRB 090926A werden zwei unterschiedliche Kurven angepasst. Die Abbildungen 4.3, 4.4 und 4.5 sind jeweils in eine Abbildung a und in eine Abbildung b unterteilt. Der Unterschied in den zwei zusammengehörigen Abbildungen verdeutlicht das in den Kapiteln 3.5 und 3.6 angesprochene Problem, dass zu wenig Zeitintervalle mit Null detektierten Ereignissen berücksichtigt werden. In der jeweiligen Abbildung a) ist die Kurve durch den Eintrag der Histogrammklasse mit Null Ereignissen pro Zeitintervall angepasst worden. In der jeweiligen Abbildung b) hingegen findet die Anpassung nicht durch die Klasse 0 statt.

GRB 090902B ist in Pixel 3282 zu finden. Das Histogramm in Abbildung 4.2 zeigt, dass sowohl ein Zeitintervall mit 31 detektierten Ereignissen, als auch ein Zeitintervall mit 33 detektierten Ereignissen vorliegt. Für diesen GRB sollen laut Tabelle 4.1 etwa 30 Photonen detektiert worden sein. Es ist zu beachten, dass die benachbarten Zeitintervalle sich um 50 s überlappen. Daher ist davon auszugehen, dass 31 Photonen innerhalb von 50 s detektiert wurden.
Gamma-ray Burst 090902B wurde dementsprechend mit dieser Methode rekonstruiert.
Die Wahrscheinlichkeit, die 33 in einem Zeitintervall detektierten Photonen zufällig beobachtet zu haben, liegt bei $3{,}8 \cdot 10^{-29}$. Bei 4 Ereignissen liegt die Wahrscheinlichkeit bei etwa 10^{-5}.

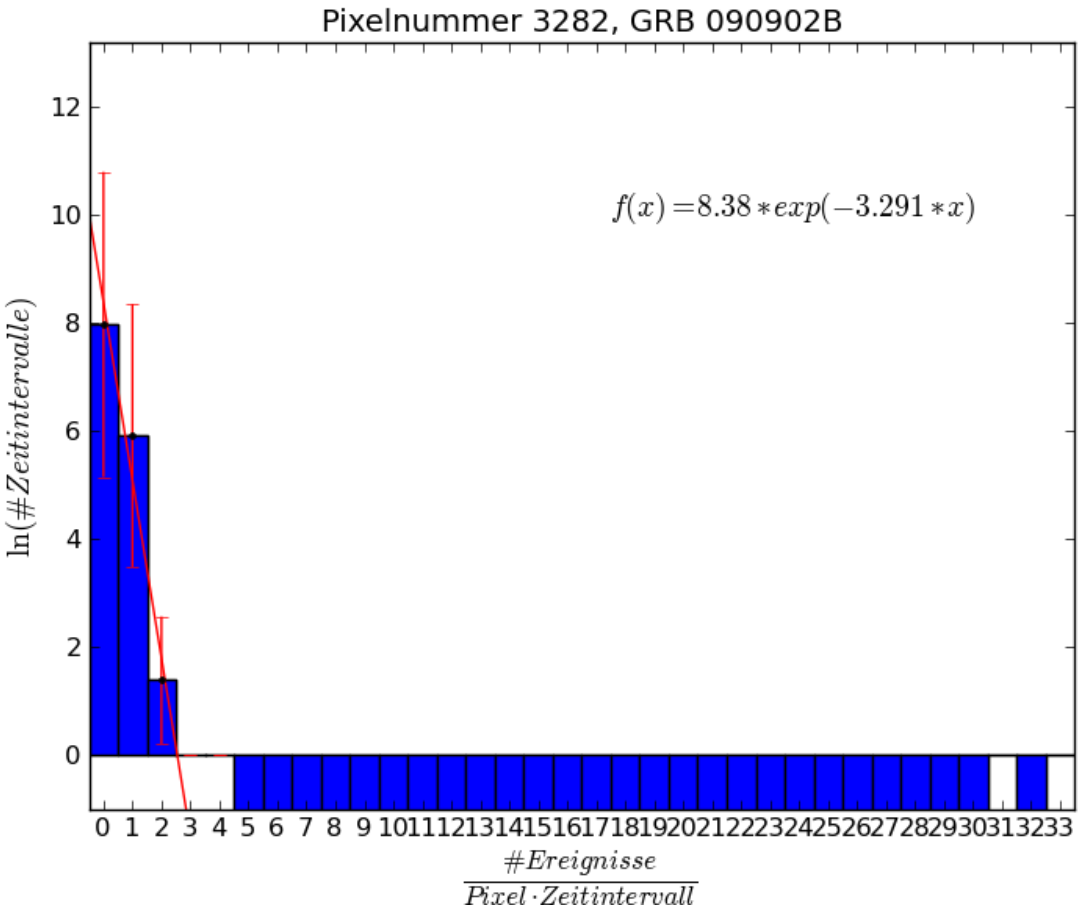
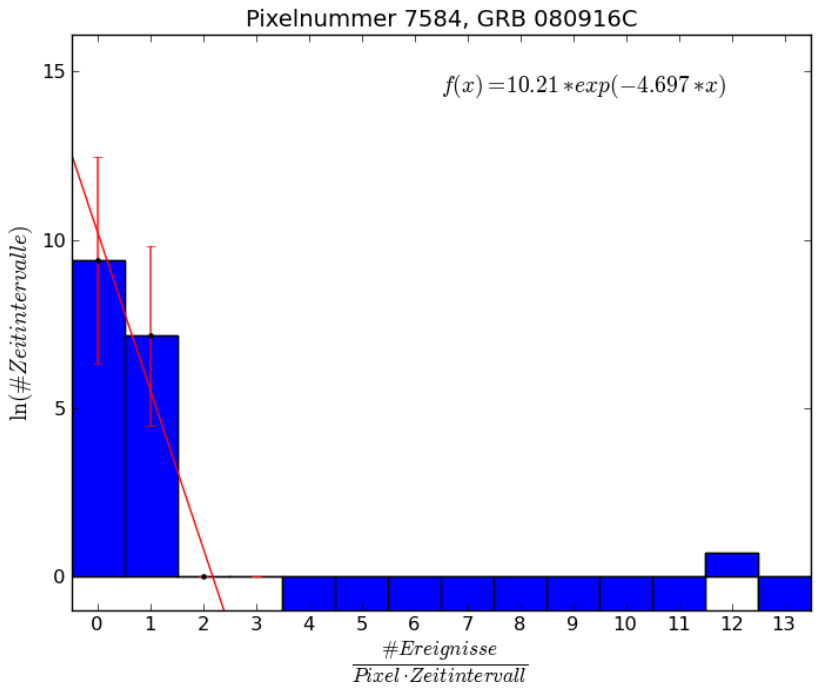
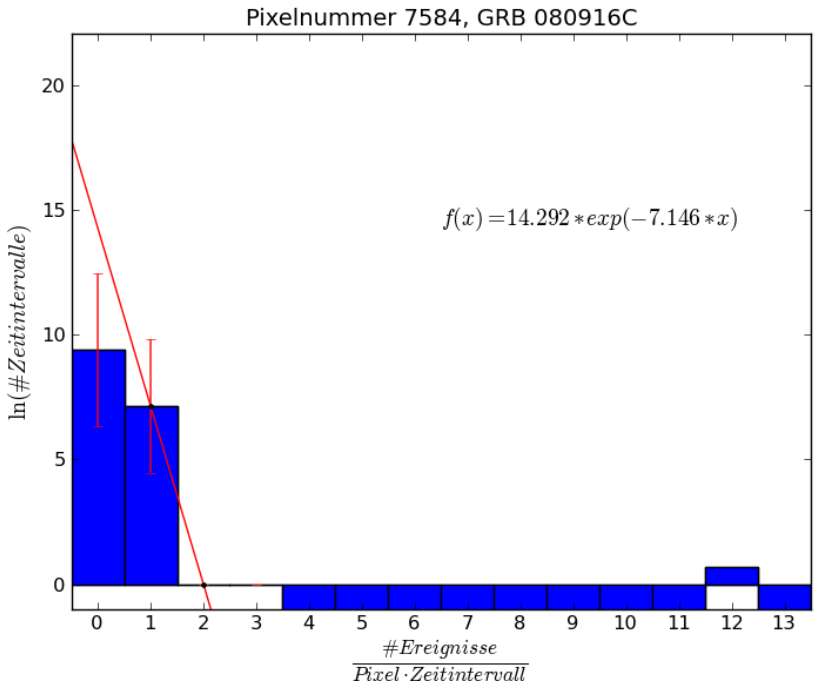


Abbildung 4.2: Angepasste Gerade an die Histogrammdaten des Pixels 3282, durch die Höhen der Klassen 0, 1 und 2. Dieses Pixel enthält GRB 090902B.

GRB 080916C ist in Pixel 7584 zu finden. Die Histogramme in Abbildung 4.3 zeigen, dass es zwei Zeitintervalle mit 12 detektierten Ereignissen gibt. All diese Ereignisse wurden innerhalb von 50 s detektiert. Nur so können zwei Zeitintervalle mit der selben Anzahl an Photonen vorliegen. Für diesen GRB sollen laut Tabelle 4.1 13 hochenergetische Photonen detektiert worden sein. Diese Diskrepanz von einem Photon kann durch die gemachten Schnitte erklärt werden. Es ist möglich, dass dieses Photon z.B. auf Grund der Einschränkung der Ereignisklasse nicht mit in den analysierten Datensatz aufgenommen wurde. Es besteht auch die Möglichkeit, dass dieses Photon innerhalb der *afterglow*-Phase gemessen wurde und damit in einem späteren Zeitintervall verzeichnet ist. Auch der Gamma-ray Bursts 080916C wird von dieser Analysemethode gefunden. Die inversen Rate α der Exponentialfunktion ist im Histogramm mit Anpassung durch die Klasse 0 (siehe Abb. 4.3(a)) um einen Faktor 1,5 kleiner als in dem Histogramm ohne Anpassung durch die Klasse 0. Dadurch wird das zuvor beschriebene Problem der zu wenigen Nullen deutlich. Wären alle Zeitintervalle, in denen das Pixel von Fermi beobachtet wurde, mit in das Histogramm aufgenommen worden, so dürfte nur ein geringer Unterschied zwischen den beiden inversen Raten vorliegen.
Die Wahrscheinlichkeit, die 12 in einem Zeitintervall detektierten Photonen zufällig beobachtet zu haben, liegt bei $3{,}4 \cdot 10^{-24}$. Bei 3 Ereignissen pro Zeinintervall liegt die Wahrscheinlichkeit bei etwa 10^{-5}.



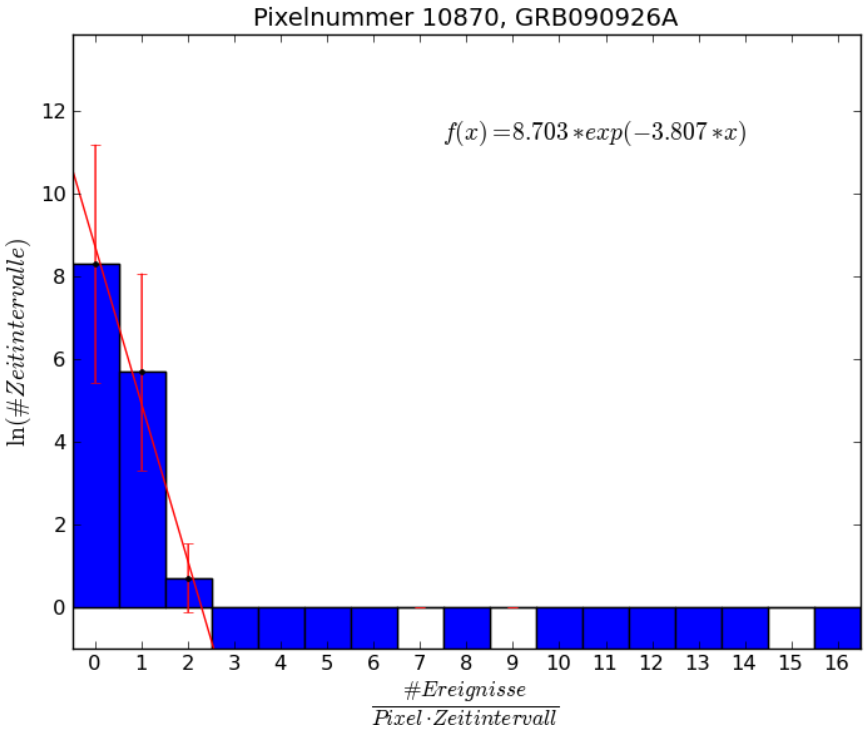
(a)



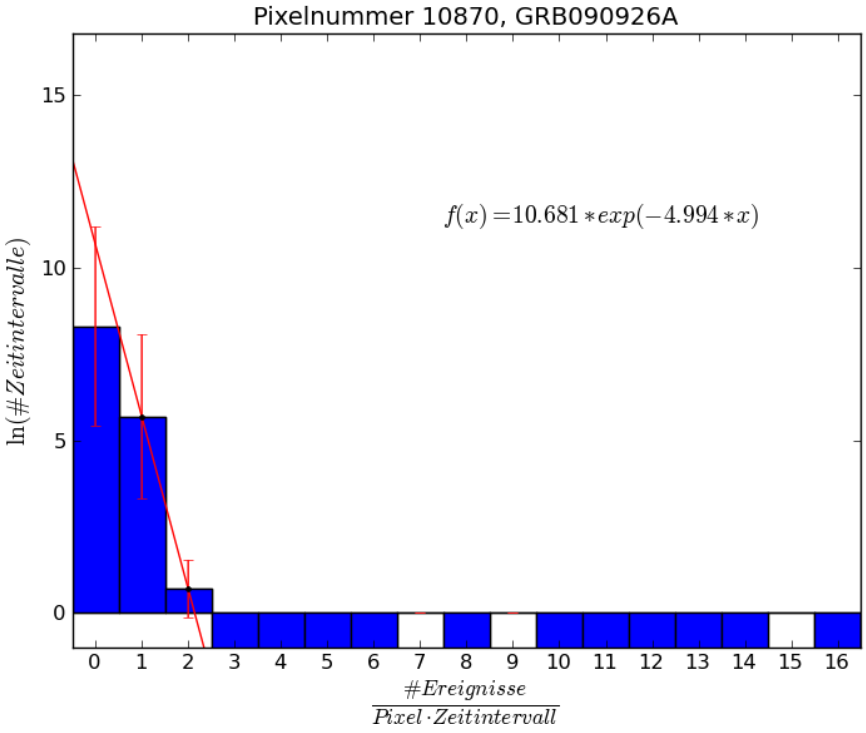
(b)

Abbildung 4.3: Angepasste Geraden an die Histogrammdaten des Pixels 7584. Dieses Pixel enthält GRB 080916C. (a) Anpassung einer Funktion durch die Höhen der Klassen 0, 1 und 2. (b) Anpassung einer Funktion durch die Höhen der Klassen 1 und 2.

GRB 090926A ist in Pixel 10870 zu finden. In den Histogrammen in Abbildung 4.4 ist sichtbar, dass es ein Zeitintervall mit 7 Photonen, eines mit 9 detektierten Ereignissen und ein Zeitintervall mit 15 detektierten Ereignissen gibt. Bouvier (Bouvier, 2010) hat in seiner Veröffentlichung angegeben, dass bei diesem Gamma-ray Burst etwa 50 hochenergetische Photonen detektiert wurden. In der Veröffentlichung von Swenson et al. (C.A. Swenson et al., 2010) wird GRB 090926A beschrieben. Es wird angegeben, dass 20 Photonen mit Energien im GeV-Bereich innerhalb der ersten 300 s detektiert wurden. Desweiteren ist der Veröffentlichung zu entnehmen, dass die *afterglow*-Phase länger als 11 Tage andauerte. Somit ist zu vermuten, dass ein Großteil der 50 von Bouvier genannten Photonen dieses Gamma-ray Bursts innerhalb der *afterglow*-Phase detektiert wurden. In dem Fall würden sie innerhalb größerer Zeitabstände als 100 s, also nicht innerhalb eines Zeitintervalls, gemessen worden sein. Die Wahrscheinlichkeit, dass zufällig ein Zeitintervall mit 15 detektierten Ereignissen auftritt, kann mit $1{,}4 \cdot 10^{-24}$ abgeschätzt werden. Dies ergibt sich aus Gleichung 2.6 in Kapitel 2.2 und der Analyse dieses Pixels mit dem in Kapitel 3.7 beschriebenen Programm „auswertung.py“. Desweiteren sind diese 15 Photonen und die 7+9=16 Photonen aus den anderen Zeitintervallen konsistent mit der Aussage aus (C.A. Swenson et al., 2010), dass 20 Photonen innerhalb der ersten 300 s gemessen wurden. Auch GRB 090926A wurde mit dieser Methode erkannt. Wie schon bei GRB 080916C wird auch in diesem Fall der Unterschied zwischen den beiden Anpassungsmethoden deutlich. In der Anpassung aus Histogramm 4.4(a) ist die inverse Rate um einen Faktor 1,3 geringer als bei der in Histogramm 4.4(b) verwendeten Methode.
Die Wahrscheinlichkeit, die 15 in einem Zeitintervall detektierten Photonen zufällig beobachtet zu haben, liegt bei $1{,}4 \cdot 10^{-24}$. Bei 4 Ereignissen pro Zeinintervall liegt die Wahrscheinlichkeit bei etwa 10^{-5}.



(a)

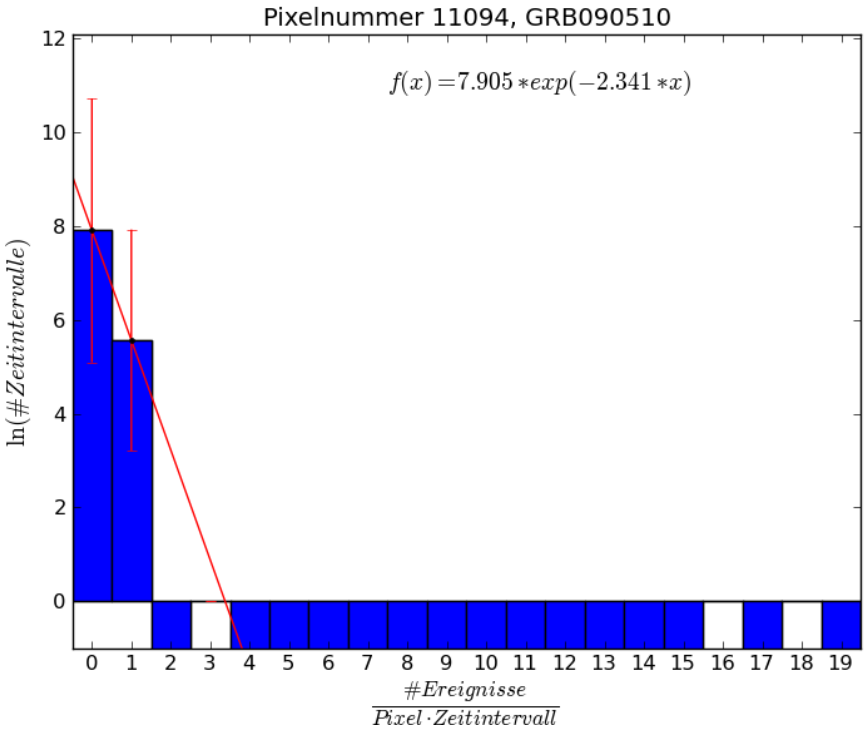


(b)

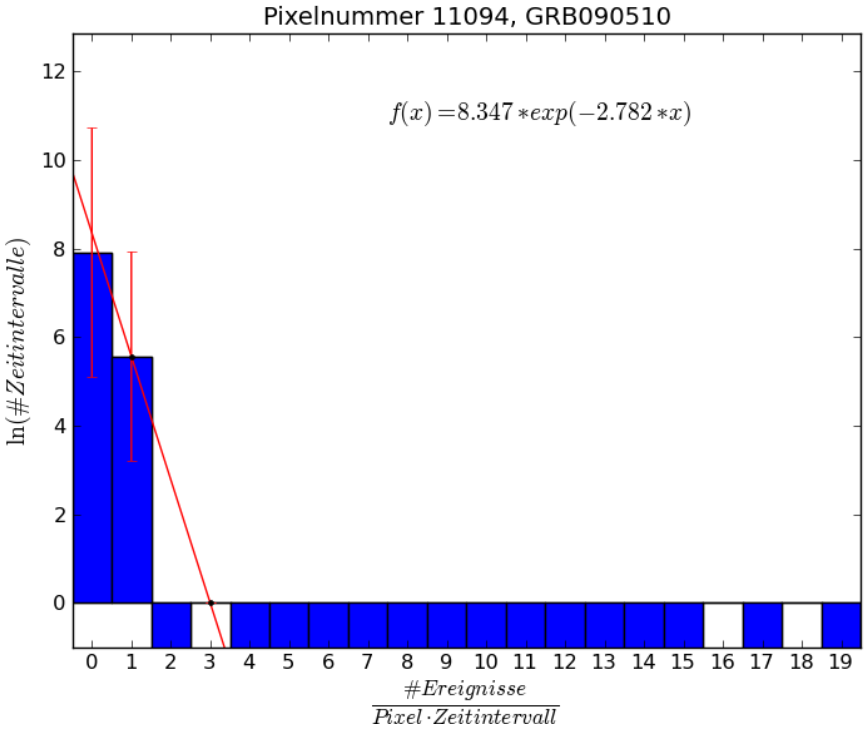
Abbildung 4.4: Angepasste Geraden an die Histogrammdaten des Pixels 10870. Dieses Pixel enthält GRB 090926A. (a) Anpassung einer Funktion durch die Höhen der Klassen 0, 1 und 2. (b) Anpassung einer Funktion durch die Höhen der Klassen 1 und 2.

In Abbildung 4.5 sind die Histogramme zu GRB 090510 in Pixel 11094 zu sehen. Es wird deutlich, dass ein Zeitintervall mit 16 und eines mit 18 detektierten Ereignissen existiert. Es ist zu vermuten, dass mindestens 16 der Photonen innerhalb von 50 s detektiert wurden. Zu diesem Gamma-ray Burst hat Bouvier (Bouvier, 2010) angegeben, dass 20 hochenergetische Photonen detektiert wurden. Auch in diesem Fall kann die Differenz zu den erwarteten 20 Ereignissen durch die gemachten Schnitte und die *afterglow*-Phase erklärt werden. Die Ergebnisse aus dieser Analyse sind mit den Angaben zu GRB 09510 konsistent, sodass dieser Burst ebenfalls als gefunden angesehen werden kann. Auch in diesem Fall soll noch einmal auf den Unterschied in der inversen Rate der beiden Anpassungsmethoden in den Abbildungen 4.5(a) und 4.5(b) hingewiesen werden.
Die Wahrscheinlichkeit, die 18 in einem Zeitintervall detektierten Photonen zufällig beobachtet zu haben, liegt bei $4{,}0 \cdot 10^{-18}$. Bei 6 Ereignissen pro Zeinintervall liegt die Wahrscheinlichkeit bei etwa 10^{-5}.

In der weiteren Auswertung wird auf die angepasste Funktion eines Pixels zurückgegriffen. Es muss beachtet werden, dass die Anpassung auf Grund der fehlenden Einträge in der Histogrammklasse 0 nur sehr ungenau ist. Dadurch werden die Ergebnisse verfälscht und dienen lediglich als grobe Abschätzung, nicht jedoch als exakter Wert.



(a)



(b)

Abbildung 4.5: Angepasste Geraden an die Histogrammdaten des Pixels 11094. Dieses Pixel enthält GRB 090510. (a) Anpassung einer Funktion durch die Höhen der Klassen 0 und 1. (b) Anpassung einer Funktion durch die Höhen der Klassen 1 und 3.

4.3 Helle Quellen

Neben Gamma-ray Bursts werden mit dieser Analysemethode auch andere Quellen von Gammastrahlung sichtbar. So kann beispielsweise in Pixel 6786 der Krebsnebel (siehe Abb. 4.7) und in Pixel 9950 der Quasar 3C454.3 (siehe Abb. 4.6) ausgemacht werden.
Der Quasar 3C454.3 würde bei der Analyse als „interessantes Pixel“ ausgegeben werden, da die nicht angepassten Klassenhöhen über den erwarteten Werten der angepassten Kurve liegen (siehe Abschnitt 3.6). Im Fall des Quasars liegen die Werte der Klassen 3, 4, 5 und 6 über der angepassten Kurven. Zeitintervalle mit mehr als 6 Ereignissen wurden nicht festgestellt. Der Krebsnebel hingegen würde nicht als „interessantes Pixel“ identifiziert werden, obwohl es viel mehr Zeitintervalle mit detektierten Ereignissen aufweist, als das Hintergrundpixel (siehe Abschnitt 4.1). Bei dem Krebsnebel wurden innerhalb eines Zeitintervalls maximal 3 Ereignisse festgestellt.
Im Vergleich zu Hintergrundpixeln (siehe Abb. 4.1) sind die Histogrammklassen sowohl bei dem Quasar als auch bei dem Krebsnebel höher.

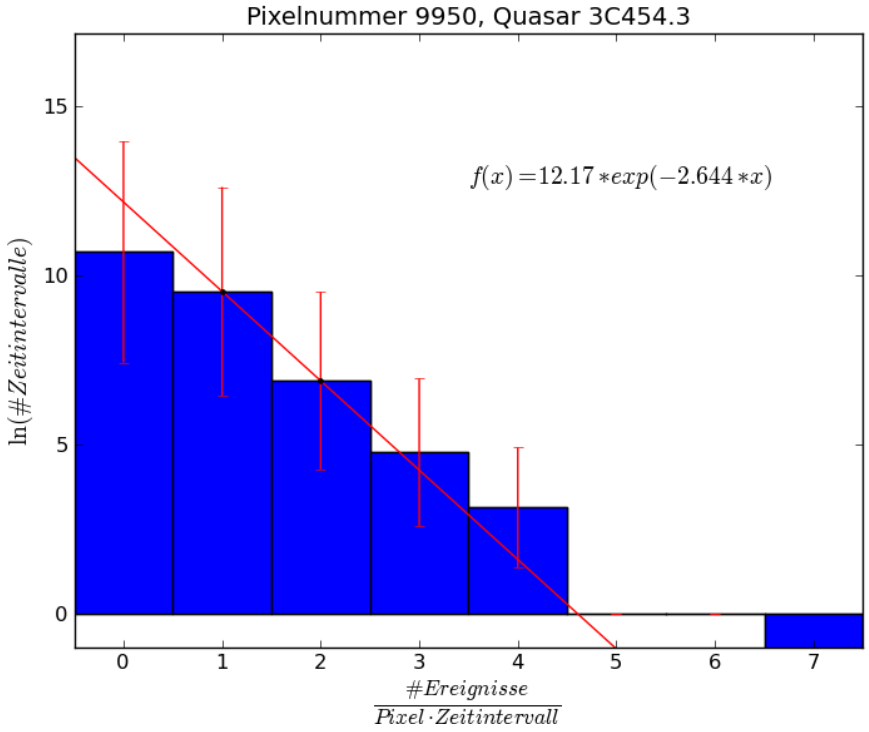


Abbildung 4.6: Histogrammdaten des Pixels 9950, in dem sich der Quasar 3C454.3 befindet.

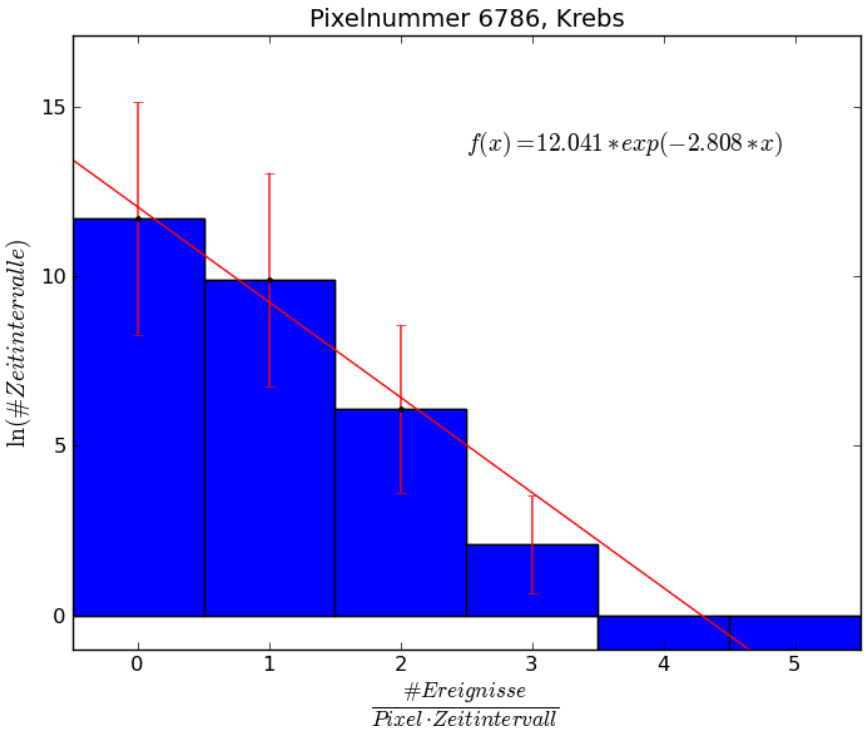


Abbildung 4.7: Histogrammdaten des Pixels 6786, das den Krebsnebel beinhaltet.

4.4 Interessante Pixel

Mit dem Programm aus Kapitel 3.6 wurde eine Himmelskarte (siehe Abschnitt 2.4.2) erstellt, die die inverse Rate α als Eintrag in jedem Pixel enthält. Noch nicht ausgewertete Pixel haben den Wert 6 erhalten, um sie von den restlichen Pixeln unterscheiden zu können. Diese Himmelskarte ist in Abbildung 4.8 zu finden. Es fällt auf, dass die inverse Rate von der beobachteten Region im Himmel abhängt.
Bei einer weiteren Analyse ist zu erwarten, dass in der galaktischen Ebene mehr Ereignisse detektiert werden. Die inverse Rate sollte dementsprechend sinken.

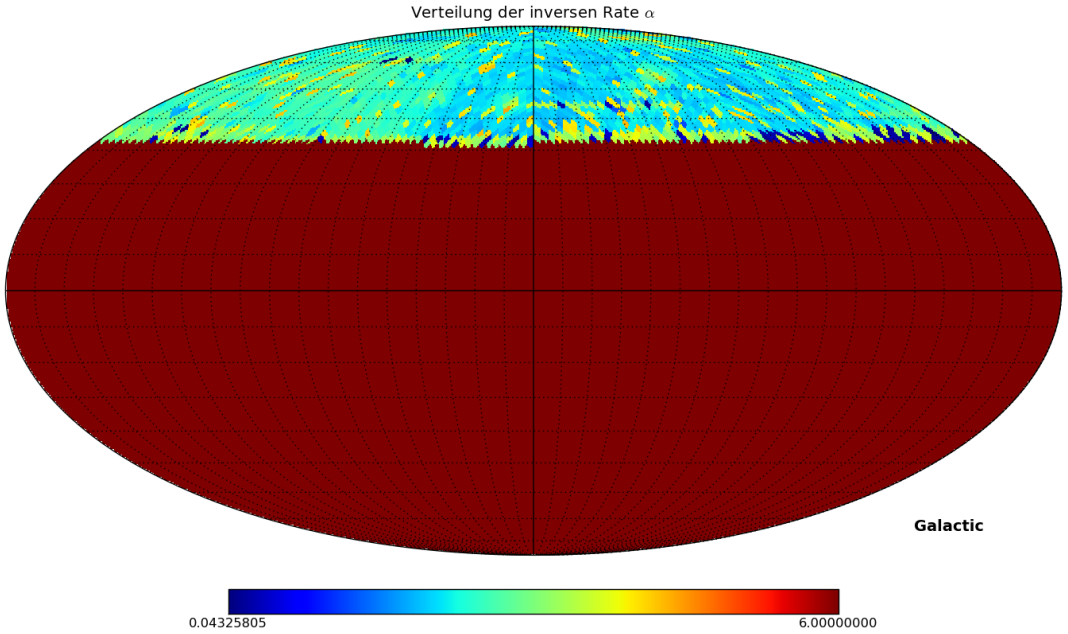


Abbildung 4.8: Himmelskarte in galaktischen Koordinaten mit der inversen Rate α (siehe Abschnitt 3.6) zu jedem Pixel. Pixel mit dem Wert 6 sind noch nicht untersucht worden.

Desweiteren wurde nach auffälligen Pixeln gesucht. In Abbildung 4.9 ist Pixel 54 als das einzige Pixel eingetragen, dessen Klassenhöhen ab Klasse 4 über den zugehörigen Funktionwerten der angepassten Kurve liegen.

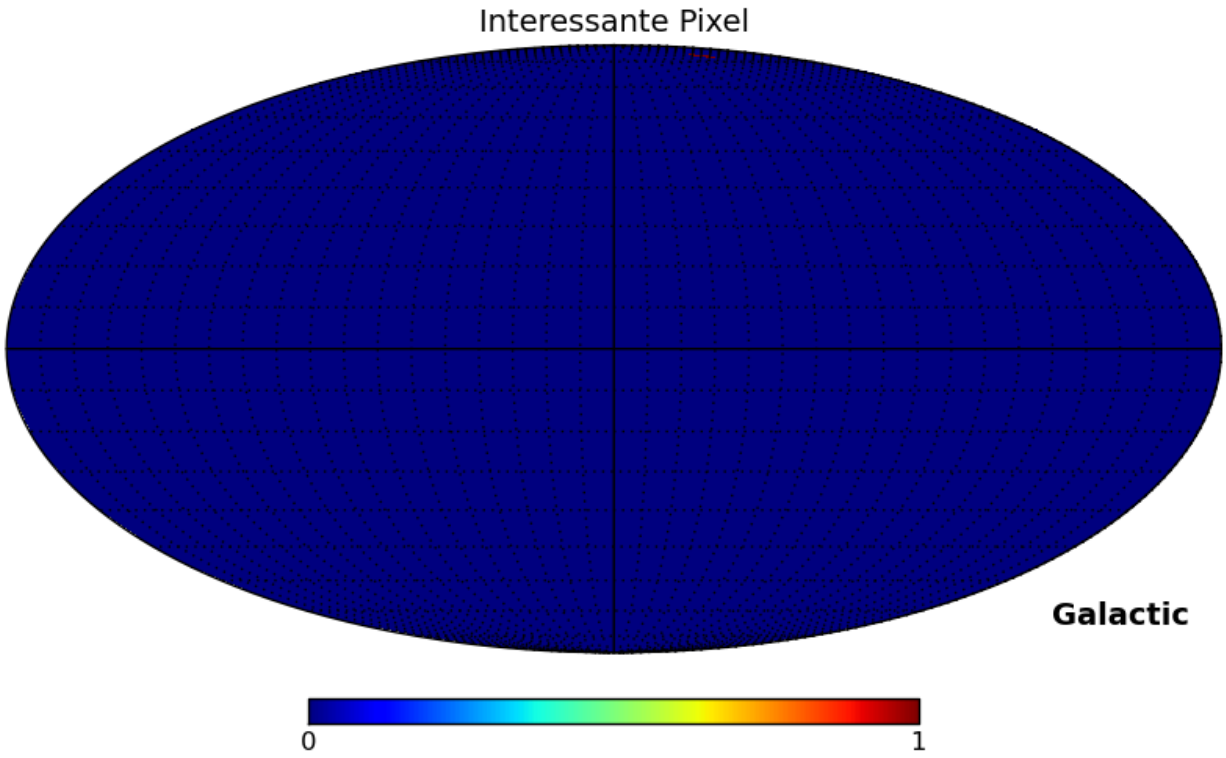


Abbildung 4.9: Himmelskarte mit dem einzigen als „interessant“ klassifizierten Pixel, Pixel 54.

In Pixel 54 liegt laut der Datenbank SIMBAD die Quelle 1FGL J1224.7+2121. Dies ist der Quasar 4C 21.35. In den Histogrammdaten (siehe Abb. 4.10) ist zu sehen, dass maximal 4 Ereignisse in einem Zeitintervall detektiert wurden.

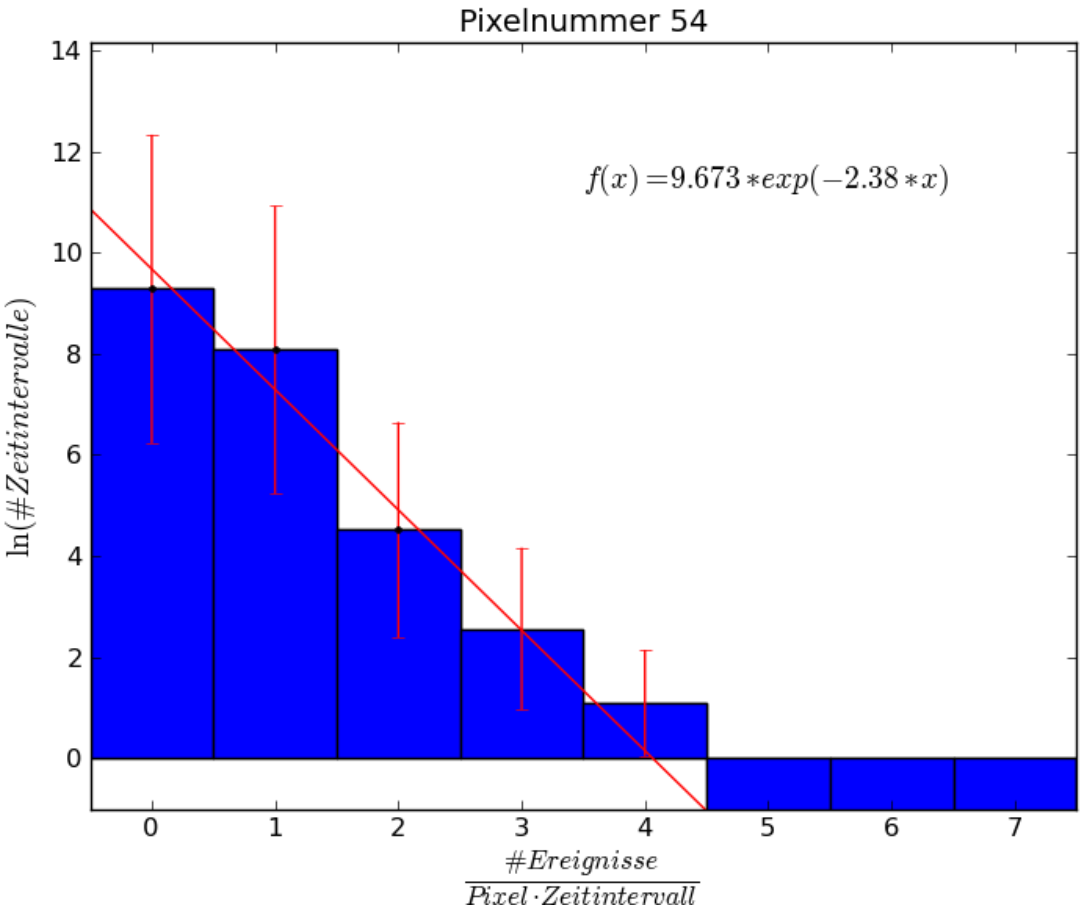


Abbildung 4.10: Histogrammdaten zu Pixel 54.

Innerhalb der untersuchten 2000 Pixel ist kein unbekanntes, transientes Ereignis detektiert worden. Dies bedeutet jedoch nicht, dass kein primordiales Schwarzes Loch evaporiert ist. Zum einen konnten bisher nicht alle Daten ausgewertet werden und zum anderen ist das Gesichtsfeld des Fermi-Teleskops begrenzt. Es werden nur ca. 20% des Himmels auf einmal beobachtet, daher ist es möglich, dass die Evaporation eines primordialen Schwarzen Loches stattfand und nicht beobachtet wurde.

Kapitel 5

Auswertung

In diesem Kapitel soll eine Abschätzung auf die Massendichte primordialer Schwarzer Löcher gemacht werden. Es werden mehrere Näherungen vorgenommen, die dazu führen, dass das Ergebnis nur ein grober Richtwert und keine exakte Grenze ist. Auch wurde diese Auswertung nur für einen kleinen Bereich des Himmels durchgeführt. Eine Untersuchung des gesamten Himmels würde das Ergebnis verbessern.

5.1 Poisson-Verteilung

Betrachtet man ein Pixel und sieht das Auftreten von einer gewissen Anzahl m an Hintergrundereignissen in einem Zeitintervall als Erfolg an, so handelt es sich um ein Bernoulli-Experiment.
Die Näherung der Poissonstatistik (siehe Abschnitt 2.2) ist für $m > 7$ Ereignisse pro Zeitintervall gerechtfertigt, da, wie in Kapitel 4 gesehen, die Wahrscheinlichkeit dort ein Ereignis pro Zeitintervall zu beobachten, kleiner als 10^{-5} ist. Sei N die Anzahl der Zeitintervalle, die Fermi das jeweilige Pixel beobachtet hat, und n die Anzahl der Zeitintervalle in denen m Ereignisse beobachtet wurden, dann ist sowohl die Forderung $N \gg n$ als auch $p \ll 1$ erfüllt. Die Wahrscheinlichkeit für das Auftreten von m Ereignissen pro Zeitintervall ist demnach poissonverteilt und folgt Gleichung 2.5.

Da der Erwartungswert der Poissonverteilung nicht bekannt ist, wird er über den Funktionswert f(m) der angepassten Kurve f an der Stelle m abgeschätzt. Dieser abgeschätzte Wert soll im Folgenden mit λ bezeichnet werden. Die Funktion f wurde in Kapitel 3.6 für jedes Pixel einzeln ermittelt.

Bei kleinen Werten für λ gilt die Näherung aus Glg. 2.6 und die Wahrscheinlichkeit kann mit dem Schätzwert angenähert werden.

5.2 Reichweite

Die Evaporation eines primordialen Schwarzen Loches kann auf Grund der technischen Einschränkungen des Fermi-Teleskops nur bis auf eine gewisse Reichweite nachgewiesen werden. Um diese zu bestimmen, kann der folgende Ansatz gemacht werden:

Die Fluenz $\phi_{\mathrm{PBH}} = \mathrm{E}_{\mathrm{PBH}}/\mathrm{A}_{\mathrm{PBH}}$ eines evaporierenden primordialen Schwarzen Loches kann über seine Masse abgeschätzt werden. Die Energie $\mathrm{E}_{\mathrm{PBH}}$, die bei der Evaporation frei wird, beträgt $\mathrm{E}_{\mathrm{PBH}} = \mathrm{m}_{\mathrm{PBH}} \cdot \mathrm{c}^2$. Da allerdings nur ein Anteil η in Photonen übertragen wird, folgt $\mathrm{E}_{\mathrm{PBH},\gamma} = \eta \cdot \mathrm{m}_{\mathrm{PBH}} \cdot \mathrm{c}^2$. Der Anteil an Energie, der in Form von Photonen abgestrahlt wird, ist temperaturabhängig und kann daher nicht angegeben werden. Aus diesem Grund soll die Grenze für die drei Fälle

1. konservativ: η =0,1
2. optimistisch: η =0,5
3. unrealistisch: η =1

berechnet werden.

Die Fläche, auf die sich die abgestrahlten Photonen nach der Entfernung d verteilen, beträgt $\mathrm{A}_{\mathrm{PBH}} = 4\pi \cdot \mathrm{d}^2$. Daraus ergibt sich eine Fluenz:

$$\Phi_{PBH} = \frac{E_{PBH}}{A_{PBH}} = \frac{\eta \cdot m_{PBH} \cdot c^2}{4\pi \cdot d^2} \tag{5.1}$$

Zusätzlich kann die Fluenz für ein detektiertes transientes Ereignis abgeschätzt werden. In Kapitel 3 wurde beschrieben, dass die Energie der in dieser Analyse verwendeten Photonereignisse auf $E_\gamma \geq 1\,\mathrm{GeV}$ festgelegt wurde. Bei der Evaporation eines primordialen Schwarzen Loches ist zu erwarten, dass viele Photonen innerhalb eines Zeitintervalls detektiert werden. Nun ist eine Wahrscheinlichkeit P zu wählen, die so klein ist, dass ein Zeitintervall mit m Einträgen nicht mehr als zufällig bezeichnet werden kann. In dieser Arbeit wurde diese Wahrscheinlichkeit auf $P = 10^{-5}$ festgesetzt. Nach Glg. 2.6 entspricht diese genau dem Funktionswert $f(n_\gamma) = \lambda$. Für jede angepasste Kurve wurde in Abschnitt 3.7 das entsprechende $n_\gamma(\lambda = 10^{-5})$ ausgerechnet. Über eine Mittelung aller ausgewerteten Pixel kommt man auf einen Wert von $\overline{n_\gamma}(\lambda = 10^{-5}) \approx 7{,}18$.

Die gesamte in diesem Zeitintervall gemessene Energie würde demnach $E_{\mathrm{trans}} = n_\gamma \cdot E_\gamma \geq 7.18\,\mathrm{GeV}$ betragen.

Zur Berechnung der Fluenz ist desweiteren die effektive Fläche des Teleskops nötig. Sie beträgt bei Photonen der *Source*-Klasse mit Energien $E_\gamma \geq 1\,\mathrm{GeV}$ mehr als $A_{\mathrm{eff}} = 8000\,\mathrm{cm}^2$. Dies kann Abbildung 5.1 entnommen werden.

Damit ergibt sich eine Fluenz von

$$\Phi_{theor.} = \frac{E_{trans}}{A_{eff}} \geq \frac{n_\gamma \cdot E_\gamma}{A_{eff}} \approx 9 \cdot 10^{-4}\,\frac{\mathrm{GeV}}{\mathrm{cm}^2} \tag{5.2}$$

Vergleicht man nun die Fluenz, die bei der Evaporation eines primordiale Schwarzen Loches gemessen werden würde (siehe Glg. 5.1) mit der theoretisch berechneten (siehe Glg. 5.2), so kann durch Lösen der Ungleichung eine obere Grenze auf die Entfernung gesetzt werden.

$$d \leq \sqrt{\frac{\eta \cdot m_{PBH} \cdot c^2 \cdot A_{eff}}{4\pi \cdot n_\gamma \cdot E}} \tag{5.3}$$

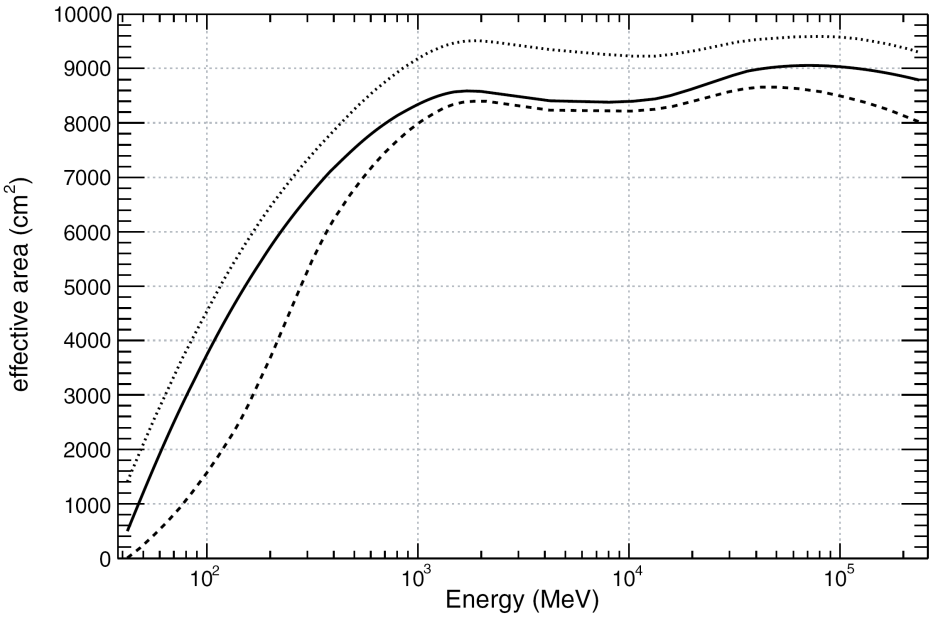


Abbildung 5.1: Effektive Fläche des LAT aufgetragen gegen die Energie für die diffuse (gestrichelte Kurve), source (durchgezogene Linie) und transient-Ereignisklasse (gepunktete Kurve) (Atwood et al., 2009).

In Kapitel 2.1.3 wurde angegeben, dass die aktuell evaporierenden primordialen Schwarzen Löcher eine Masse von $m_{PBH} \approx (5{,}7 \pm 1{,}4) \cdot 10^{14}\,g$ aufweisen. Wird die obere Grenze von $m_{PBH} \approx 7{,}1 \cdot 10^{14}\,g$ angenommen, so folgt für die drei verschiedenen Fälle eine Entfernung, bis zu der die Evaporation eines primordialen Schwarzen Loches mit dem LAT messbar ist. Mit Hilfe dieser Entfernung kann ein Volumen $V_{ges} = \frac{4}{3}\pi d^3$ berechnet werden. Da allerdings nur 2000 von 12288 Pixeln untersucht wurden, ist das Volumen um einen entsprechenden Faktor zu reduzieren. Es folgt ein Volumen $V = \frac{2000}{12288} \cdot \frac{4}{3}\pi d^3$. Die Ergebnisse sind in Tabelle 5.1 dargestellt. Es wurde mit $c \approx 299\,792\,458\,\frac{m}{s}$ gerechnet und genutzt, dass $1\,m \approx 3.24078 \cdot 10^{-17}\,pc$ und $1\,eV \approx 1{,}602 \cdot 10^{-19}\,J$ entspricht.

Tabelle 5.1: Übersicht über den Radius d und das Volumen V in dem die Evaporation eines primordialen Schwarzen Loches für verschiedene Anteile η detektiert werden kann

η	Entfernung d [pc]	Volumen V [pc^3]
0,1	$\lesssim$ 19,25	$\lesssim$4866
0,5	$\lesssim$ 43,05	$\lesssim$54402
1	$\lesssim$ 60,88	$\lesssim$153872

Für eine weitere Analyse ist jedoch zu beachten, dass Pixel, in denen ein Gamma-ray Burst oder eine andere Gammaquelle lokalisiert wurde, nicht zu der Analyse beitragen können. Zeitintervalle, in denen Photonen aus einer solchen Quelle gezählt wurden, können mit dieser Analyse nicht von denen aus der Evaporation eines primordialen Schwarzen Loches unterschieden werden. Somit müsste bei einer Analyse des gesamten Himmels auch die galaktische Ebene von der Auswertung ausgeschlossen werden.

5.3 Massendichte

Mit Hilfe des Volumens, in dem die Evaporation eines primordialen Schwarzen Loches detektiert werden kann, kann eine obere Grenze auf die Anzahldichte der primordialen Schwarzen Löcher ausgerechnet werden.

$$\frac{n_{PBH}}{t} \leq \frac{1}{V \cdot \Delta T}$$

ΔT gibt die Beobachtungszeit von 101 808 985 s an. Es ist allerdings zu berücksichtigen, dass das Gesichtsfeld des LAT nur etwa 20% des Himmels entspricht. Setzt man voraus, dass jedes Pixel gleich viel Beobachtungszeit bekommt, so wird ein Pixel nur 20% der Zeit betrachtet. Daraus folgt für die beobachtete Zeit $\Delta \mathrm{T} = 20\,361\,797\,\mathrm{s}$. Mit $t = t_{PBH}$ als Lebensdauer der primordialen Schwarzen Löcher, die mit der Hubblezeit zu $t_{PBH} = \frac{1}{H_0} \approx 10{,}893a \approx 5{,}1529 \cdot 10^{17}$s (für $h_0 = 0{,}9$) (siehe Abschnitt 2.1.3) abgeschätzt werden kann, folgt:

$$n_{PBH} \leq \frac{t_{PBH}}{V \cdot \Delta T}$$

Mit $\rho = \mathrm{m} \cdot \mathrm{n}$ erhält man ein Grenze auf die Massendichte. Dieses ist für die drei bereits diskutierten Fälle in Tabelle 5.2 zusammengetragen.

Tabelle 5.2: Übersicht über die berechneten Größen Radius d, Volumen V, Anzahldichte n_{PBH} und Massendichte ρ_{PBH} für verschiedene Anteile η

η	Entfernung d	Volumen V	Anzahldichte n_{PBH}	Massendichte ρ_{PBH}
0,1	$\lesssim 19.25\,\mathrm{pc}$	$\lesssim 4866\,\mathrm{pc}^3$	$\lesssim 5{,}2 \cdot 10^6\ \frac{1}{\mathrm{pc}^3}$	$\lesssim 3{,}7 \cdot 10^{21}\ \frac{\mathrm{g}}{\mathrm{pc}^3}$
0,5	$\lesssim 43.05\,\mathrm{pc}$	$\lesssim 54\,402\,\mathrm{pc}^3$	$\lesssim 465\,172\ \frac{1}{\mathrm{pc}^3}$	$\lesssim 3{,}3 \cdot 10^{20}\ \frac{\mathrm{g}}{\mathrm{pc}^3}$
1	$\lesssim 60.88\,\mathrm{pc}$	$\lesssim 153\,872\,\mathrm{pc}^3$	$\lesssim 164\,463\ \frac{1}{\mathrm{pc}^3}$	$\lesssim 1{,}2 \cdot 10^{20}\ \frac{\mathrm{g}}{\mathrm{pc}^3}$

5.4 Diskussion

In Kapitel 2.1.3 wurde die Rechnung von Overduin und Wesson (J.M. Overduin, 2004) erwähnt. Die in dieser Veröffentlichung berechnete Massendichte beträgt $\rho_{PBH} \approx 5{,}7410^{18}\ \frac{\mathrm{g}}{\mathrm{pc}^3}$. Die in dieser Analyse ermittelte Grenze auf die Massendichte liegt mit $\rho_{PBH} \lesssim 3{,}7 \cdot 10^{21}\ \frac{\mathrm{g}}{\mathrm{pc}^3}$ im konversativsten Fall deutlich unter der Massendichte von Wesson und Overduin. Es sei jedoch zu beachten, dass in der in dieser Analyse durchgeführten Rechnung viele Näherungen vergenommen wurden.
Die angepassten Funktionen, und damit auch die in dieser Rechnung verwendeten Werte für $\overline{n_\gamma}(\lambda = 10^{-5}) \approx 7{,}18$ sind nicht korrekt. Dies wurde bereits in Kapitel 4.2 gezeigt und diskutiert. Bei korrekten Werten für n_γ wäre eine gewichtete Integration an Stelle der Mittelwertbildung sinnvoller, da auf diese Weise auch Vordergrundquellen berücksichtigt werden. Zusätzlich wurde die effektive Fläche des Fermi-LAT nur grob abgeschätzt. Bei der Ermittlung einer Grenze auf die Massendichte primordialer Schwarzer Löcher am gesamten Himmel, sollte diese genauer angegeben werden. Desweiteren wurden lediglich drei mögliche Fälle für den Anteil der Energie, die bei einer Evaporation eines primordialen Schwarzen Loches in Photonen übertragen wird, berechnet. Der genaue Wert ist unbekannt.

Unter Berücksichtigung dessen, ist eine Massendichte von $\rho_{PBH} \lesssim 3{,}7 \cdot 10^{21}\ \frac{\mathrm{g}}{\mathrm{pc}^3}$ eine akzeptable Abschätzung.

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurden Fermi-LAT-Daten von 169 Wochen auf der Suche nach transienten Ereignissen ausgewertet. Dies entspricht der gesamten Zeit seit dem Start des Fermi-Satelliten im Juni 2008. Das Fermi-LAT deckt einen Energiebereich von 20 MeV bis 300 GeV ab und weist eine Totzeit von nur 26.5 µs auf, sodass dieses Teleskop geeignet ist, um transiente Ereignisse im hochenergetischen Bereich zu messen. Im Energiebereich von 1 GeV bis 300 GeV wurden in den untersuchten 169 Wochen am gesamten Himmel 6 857 031 Ereignisse detektiert. Zur Lokalisierung dieser Ereignisse wurde der Himmel in 12288 Pixel unterteilt und die Ereignisse wurden entsprechend ihrer rekonstruierten Richtung einzelnen Pixeln zugeordnet. Zwecks dieser Analyse wurden die Daten mit Hilfe verschiedener, größtenteils selbst entwickelter Programme verarbeitet. Die in einem Pixel detektierten Ereignisse wurden entsprechend ihrer Detektionszeit in einzelne 100 s lange Zeitintervalle einsortiert, sodass für jedes Zeitintervall eine Anzahl an Ereignissen ermittelt werden konnte. Diese Daten wurden histogrammiert, um nach Abweichungen von einer angepassten Exponentialfunktion zu suchen. Bis zum Ende dieser Arbeit wurden lediglich 2000 der 12288 Pixel untersucht. In diesem Bereich des Himmels ist kein Anzeichen für die Evaporation eines primordialen Schwarzen Loches gefunden worden. Dies bedeutet jedoch nicht, dass ein solches Phänomen nicht stattgefunden haben kann. Jedes Pixel wird nur ca. 20% der Zeit von dem Fermi-Teleskop beobachtet und ein solches transientes Ereignis könnte dementsprechend schlichtweg „verpasst“ worden sein. Es wurde eine grobe Abschätzung auf die Massendichte primordialer Schwarzer Löcher vorgenommen. Auf Grund einiger Näherungen und der Tatsache, dass die an die Histogrammdaten angepassten Exponentialfunktionen verfälscht sind, muss jedoch betont werden, dass diese Abschätzung aller Voraussicht nach sehr ungenau ist. Kann davon ausgegangen werden, dass die Evaporation

eines primordialen Schwarzen Loches, bei einem Anteil $\eta = 0{,}1$ der Energie, die in Photonen übergeht, in einem Volumen von $V \lesssim 4866\,\mathrm{pc}^3$ detektiert werden kann, so ergibt sich für die Massendichte:

$$\rho_{PBH} \lesssim 3{,}7 \cdot 10^{21}\,\frac{\mathrm{g}}{\mathrm{pc}^3}$$

Für die Fortsetzung der Analyse der Fermi-LAT-Daten auf der Suche nach transienten Ereignissen könnten einige Veränderungen vorgenommen werden.
Es ist sinnvoll die Refernezdaten zur Auswahl der Zeitintervalle, in denen das LAT nicht auf ein bestimmtes Pixel gerichtet war, zu erweitern. So könnte zum Beispiel ein Energiebereich von 20 MeV bis 1 GeV herangezogen werden. Dies entspricht allen bei der Analyse nicht herangezogenen Ereignissen, die das Fermi-LAT detektieren kann. Um Fehler seitens des Arbeitsspeichers zu umgehen, wäre eine Unterteilung in kleinere Energiebereiche möglich, die anschließend zusammengefasst werden würden. Dies ist realisierbar, indem das in Kapitel 3.4 beschriebene Programm auf die einzelnen Energiebereiche angewendet werden würde und die so erstellten Daten addiert würden. So würden Daten entstehen, die für jedes Zeitintervall die Gesamtzahl an Ereignissen aus dem gesamten Energiebereich enthalten.
Desweiteren würde eine Ausweitung des Zeitraums, aus dem die LAT-Daten analysiert werden, bei der Verbesserung der Analyse helfen. Auf diese Weise erhielte man größere Datensätze, in denen die Möglichkeit eines verzeichneten transienten Ereignisses besteht. Da eine Laufzeit des Fermi-Teleskops von 10 Jahren angedacht ist, ist diese Möglichkeit gegeben.
Ein weiterer Ansatz, um die Analyse zu verbessern, ist die Wahl kleinerer Zeitintervalle. So könnten diese, wie ehemals angedacht, auf 1 s mit 0.5 s Versatz verringert werden. Die Erkenntnisse, die aus der Analyse mit den aktuellen 100 s langen Zeitintervallen gewonnen werden, könnten genutzt werden. Zur weiteren Analyse würde es ausreichen, ausschließlich die interessanten Pixel mit kleineren Zeitintervallen erneut zu untersuchen.
Es ist zu erwarten, dass im Fall 1 s langer Zeitintervalle sehr viel mehr Intervalle mit Null detektierten Ereignissen auftreten. Dadurch würde die angepasste Funktion sich verändern, und die Wahrscheinlichkeit zufällig mehrere Ereignisse innerhalb eines Zeitintervalls zu beobachten, würde sinken. Sehr kurzzeitige Ereignisse, wie die Evaporation eines primordialen Schwarzen Loches, würden innerhalb weniger Zeitintervalle gemessen werden. Beim Histogrammieren dieser Daten würde ein transientes Ereignis noch deutlicher hervor treten als bisher.

Von einer Analyse des gesamten Himmels ist zu erwarten, dass sich die Grenze auf die Anzahldichte primordialer Schwarzer Löcher verbessert. Eine solche Grenze kann dazu genutzt werden, dunkle Materie auszuschließen. In der Veröffentlichung von Sofue et al. (Y. Sofue, 2008) ist beschrieben, dass für dunkle Materie eine Massendichte von $\rho_{\mathrm{dm}} \approx 0{,}38$ abgeschätzt werden kann. Dies ergibt $\rho_{\mathrm{dm}} \approx 2 \cdot 10^{31} \frac{\mathrm{g}}{\mathrm{pc}^3}$. Die für primordiale Schwarze Löcher ermittelte Massendichte liegt also um 10 Größenordnungen und der von Dunkler Materie. Primordiale Schwarze Löcher sind damit voraussichtlich als Kandidaten Dunkler Materie auszuschließen. Dies sollte in einer exakteren Rechnung für den gesamten Himmel jedoch überprüft werden.

Mit Hilfe einer Analyse der Fermi-Daten ist es möglich, im Rahmen einer Auswertung des gesamten Himmels über längere Zeiträume die Evaporation primordialer Schwarzer Löcher zu detektieren und auf diese weise Grenzen auf ihre Anzahl- bzw. Massendichte zu setzen.

Danksagung

Zunächst möchte ich Prof. Dr. Dieter Horns danken, dass er mich in seine Arbeitsgruppe aufgenommen und mich mit hilfreichen Ratschlägen unterstützt hat. Einen Dank gilt der gesamten Arbeitsgruppe, die mich sehr freundlich aufgenommen und mir geholfen hat. Mir wurde sehr viel Geduld zu teil, obwohl ich den astro-wgs02 das ein oder andere Mal außer Gefecht gesetzt habe. In diesem Sinne möchte ich mich speziell bei Alexander Gewering-Peine bedanken, der sich insbesondere in meiner Anfangszeit viel mit meinen Coumputerproblemen auseinandersetzte. Ich möchte meinem Betreuer Milton Virgilio Fernandes danken, der mir sehr weitergeholfen hat. Er hat mir viele nützliche Ratschläge gegeben und ist mit mir auf Fehlersuche gegangen, wenn ich nicht weiter wusste. Danke an meine Eltern, die mir dieses Studium ermöglichen und mich dabei unterstützen. Vielen Dank auch für das Korrekturlesen dieser Arbeit. Ein besonderer Dank gilt meinem Freund, der mich insbesondere in den stressigen letzten Wochen sehr unterstützt.

Anhang

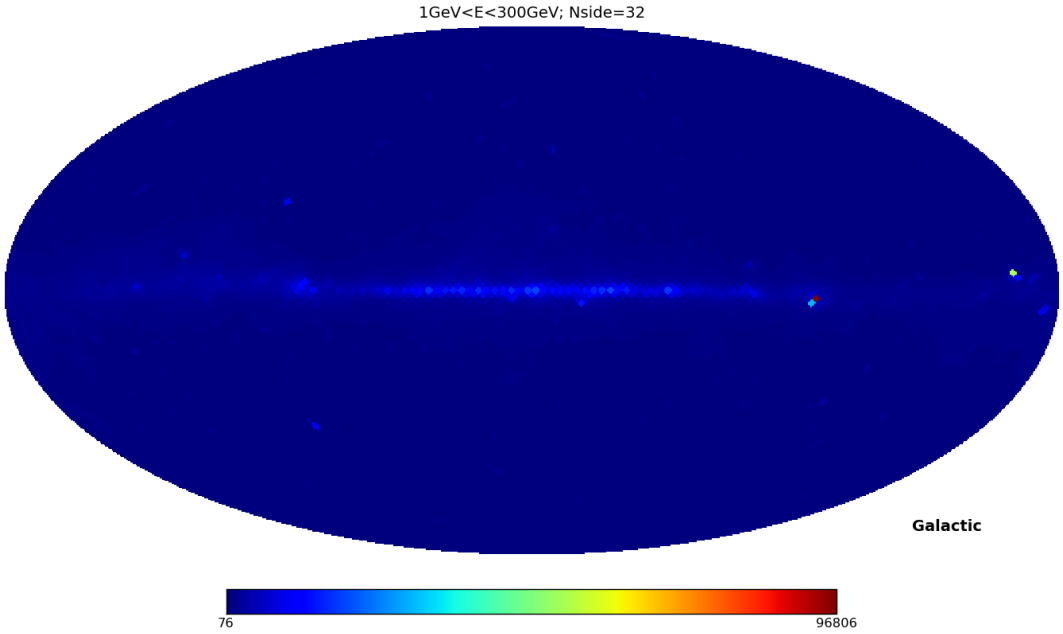


Abbildung 6.1: Himmelskarte mit den absoluten Ereignisanzahlen in jedem Pixel. Die Karte ist in 12288 Pixel unterteilt. Es werden nur Ereignisse aus dem Energiebereich von 1 GeV bis 300 GeV und aus den ersten 169 Wochen Datennahme berücksichtigt. Die Schnitte auf die Daten sind in Kapitel 3.2 beschrieben.

```
#Importiere Textdateien mit Parameterdaten und Eventdaten aus C-Programm, um diese in
Python auszuwerten
        #argv1 Parameterdaten
        #argv2 Eventdaten
        #argv3 Output-Datei


import sys
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from optparse import OptionParser
import sys
import math

##Uebergebe Start- und Endpixel an das Programm, damit Variable Pixelbereiche durchlaufen
werden koennen und kein "memoryerror" auftritt
##input params

parser=OptionParser()
parser.add_option("-s","--start",dest="start",help="Pixel mit dem die Analyse beginnen
soll")
parser.add_option("-e","--end",dest="end",help="Pixel mit dem die Analyse enden soll")
(options, args) = parser.parse_args()

## Fehlermeldung
if options.start == None:
  print "No startpixel!"
  parser.print_help()
  sys.exit(0)
if options.end == None:
  print "No endpixel!"
  parser.print_help()
  sys.exit(0)

##Input ist string, wandle fuer die Rechnung in integer um
start = int(options.start)
end = int(options.end)


#Startzeit des LAT als Referenz, da im C-Programm die Differenz aus aktuellem Zeitpunkt
und Startzeit gebildet wurde, fuer kuerzere Zahlen
LATstart=239557417
LAT177=341366402
#gesamte Laufzeit: 101808985


"""Oeffne Dateien und lese Daten ein"""

# Mit sys kann die zu oeffnende Datei als Argument eingegeben werden. So muss der Code
nicht fuer unterschiedliche Input-Dateien geaendert werden
#Nehme Werte aus der Parameter-Datei
Param = open(sys.argv[1])

#Liest erste Zeile der Datei fuer die Anzahl der Events und die Zeite Zeile fuer die
Anzahl der Pixel ein, springt automatisch in die naechste Zeile
naxis2=long(Param.readline())
healpix=long(Param.readline())
```

```
print 'Parameterdaten eingelesen'

#Nehme Daten aus der Event-Datei
Arrays = open(sys.argv[2])

#numpy Befehl genfromtxt kann 2 Zeilen aus Textdatei in zwei arrays umwandeln
# Lese die Pixelnummern der einzelnen Events in ein array ein
ipringctrl = np.genfromtxt(Arrays, delimiter=' ', dtype=long, usecols=(0))
#Beginne wieder am Anfang der Datei mit einlesen
Arrays.seek(0, 0)
#Lese die zu den Events gehoerenden Zeiten in ein array ein
arrayt = np.genfromtxt(Arrays, delimiter=' ', dtype=float, usecols=(1))  #Pro Event die
zugehoerige Zeit als Eintrag
Arrays.close()



xmax=math.ceil((LAT177-LATstart-100)/50)        #xmax --> Maximale Anzahl an
Zeitintervallen, ceil rundet auf, damit auch das letzte, unvollständige Zeitintervall
                                                eingelsen wird
for j in range(start,end):
        a= []
        for i in range(naxis2):
                if ipringctrl[i] ==j:           # Erstelle Liste a mit allen zu Pixel j
gehörigen Zeiten
                        a.append(arrayt[i])


        pta = np.zeros(xmax)                                            # Liste pat; für
jedes Zeitintervalle eine Null
        for x in range(0,xmax):                                         #x ist
Durchlaufvariable zum Verschieben des Zeitintervalls

                for k in range(len(a)):                                 # Durchsuche Liste
a nach Zeiten innerhalb des gewünschten Zeitintervalls
                        if a[k] >=x*50 and a[k] <=x*50+100:             # Zähle für jedes
Zeitintervall Anzahl an Ereignissen
                                pta[x] = pta[x]+1


        print 'Speichere Datei für Pixel {0}' .format(j)
        np.savetxt('/nfs/astrop/d4/ahirt/Jag/cpp/1GeV_subarray-pixel-'+str(j)+'.txt', pta,
delimiter='    ')       # Verändere Pfad je nach untersuchtem Energiebereich
```

```
import sys
import numpy as np
#import matplotlib
#matplotlib.use('TkAgg')
#import matplotlib.pyplot as plt
import glob
from optparse import OptionParser

#Eingabe:
#Output: Histogrammdatei

#Uebergebe Start- und Endpixel an das Programm, damit kein "memoryerror" auftritt
#input Parameter
parser=OptionParser()
parser.add_option("-s","--start",dest="start",help="Startpixel")
parser.add_option("-e","--end",dest="end",help="Endpixel ")
(options, args) = parser.parse_args()

# Fehlermeldung
if options.start == None:
  print "No startpixel!"
  parser.print_help()
  sys.exit(0)
if options.end == None:
  print "No endpixel!"
  parser.print_help()
  sys.exit(0)

#Input ist string, wandle fuer die Rechnung in integer um
start = int(options.start)
end = int(options.end)



n=[]

#Lese Daten in beiden Energiebereichen aus dem Programm GRB_Suche.py in
#Pixelweise Vorgehen, um so einem memory error vorzubeugen

for i in range(start, end):
        #Fuer Daten im Energiebereich 1GeV-300GeV
        cpphigh =0
        Pixels_high =
glob.glob('/nfs/astrop/d4/ahirt/Jag/cpp/GeV/1GeV_subarray-pixel-'+str(i)+'.txt')
        # Lade Dateipfad
        cpphigh = np.loadtxt(Pixels_high[0], dtype=np.float, delimiter='    ')
                        # Lese Daten aus Datei ein

        #Fuer Daten im Energiebereich 100MeV-400MeV
        cpplow =0
        Pixels_low =
glob.glob('/nfs/astrop/d4/ahirt/Jag/cpp/MeV/MeV_subarray-pixel-'+str(i)+'.txt')
        cpplow = np.loadtxt(Pixels_low[0], dtype=np.float, delimiter='    ')


        xmax = len(cpphigh)             # xmax entspricht Anzahl der Zeitintervall

#Suche nach Block aus 33 Nullen (30min) in beiden Energiebereichen
```

```
        for m in range(0, xmax-32):                    #xmax-32, damit nach 33
aufeinander folgenden Nullen gesucht werden kann
                if cpphigh[m] == 0 and cpplow[m] == 0 and cpplow[m+1] == 0 and
cpphigh[m+1] == 0 and cpplow[m+2] == 0 and cpphigh[m+2] == 0 and cpplow[m+3] == 0 and
cpphigh[m+3] == 0 and cpplow[m+4] == 0 and cpphigh[m+4] == 0 and cpplow[m+5] == 0 and
cpphigh[m+5] == 0 and cpplow[m+6] == 0 and cpphigh[m+6] == 0 and cpplow[m+7] == 0 and
cpphigh[m+7] == 0 and cpplow[m+8] == 0 and cpphigh[m+8] == 0 and cpplow[m+9] == 0 and
cpphigh[m+9] == 0 and cpplow[m+10] == 0 and cpphigh[m+10] == 0 and cpplow[m+11] == 0 and
cpphigh[m+11] == 0 and cpphigh[m+12] == 0 and cpplow[m+12] == 0 and cpplow[m+13] == 0 and
cpphigh[m+13] == 0 and cpplow[m+14] == 0 and cpphigh[m+14] == 0 and cpplow[m+15] == 0 and
cpphigh[m+15] == 0 and cpplow[m+16] == 0 and cpphigh[m+16] == 0 and cpplow[m+17] == 0 and
cpphigh[m+17] == 0 and cpplow[m+18] == 0 and cpphigh[m+18] == 0 and cpphigh[m+19] == 0 and
cpplow[m+19] == 0 and cpphigh[m+20] == 0 and cpplow[m+20] == 0 and cpphigh[m+21] == 0 and
cpplow[m+21] == 0 and cpphigh[m+22] == 0 and cpplow[m+22] == 0 and cpplow[m+23] == 0 and
cpphigh[m+23] == 0 and cpplow[m+24] == 0 and cpphigh[m+24] == 0 and cpplow[m+25] == 0 and
cpphigh[m+25] == 0 and cpplow[m+26] == 0 and cpphigh[m+26] == 0 and cpplow[m+27] == 0 and
cpphigh[m+27] == 0 and cpplow[m+28] == 0 and cpphigh[m+28] == 0 and cpphigh[m+29] == 0 and
cpplow[m+29] == 0 and cpphigh[m+30] == 0 and cpplow[m+30] == 0 and cpphigh[m+31] == 0 and
cpplow[m+31] == 0 and cpphigh[m+32] == 0 and cpplow[m+32] == 0 and cpphigh[m+33] == 0 and
cpplow[m+33] == 0:

                #Wandle aufeinanderfolgende Nullen in den GeV-Daten in -1 um, damit die
Laenge des Intervalls nicht beeinflusst wird und die Eintraege nicht in den
Histogrammdaten auftauchen

                        cpphigh[m] = -1
                        cpphigh[m+1] = -1
                        cpphigh[m+2] = -1
                        cpphigh[m+3] = -1
                        cpphigh[m+4] = -1
                        cpphigh[m+5] = -1
                        cpphigh[m+6] = -1
                        cpphigh[m+7] = -1
                        cpphigh[m+8] = -1
                        cpphigh[m+9] = -1
                        cpphigh[m+10] = -1
                        cpphigh[m+11] = -1
                        cpphigh[m+12] = -1
                        cpphigh[m+13] = -1
                        cpphigh[m+14] = -1
                        cpphigh[m+15] = -1
                        cpphigh[m+16] = -1
                        cpphigh[m+17] = -1
                        cpphigh[m+18] = -1
                        cpphigh[m+19] = -1
                        cpphigh[m+20] = -1
                        cpphigh[m+21] = -1
                        cpphigh[m+22] = -1
                        cpphigh[m+23] = -1
                        cpphigh[m+24] = -1
                        cpphigh[m+25] = -1
                        cpphigh[m+26] = -1
                        cpphigh[m+27] = -1
                        cpphigh[m+28] = -1
                        cpphigh[m+29] = -1
                        cpphigh[m+30] = -1
                        cpphigh[m+31] = -1
                        cpphigh[m+32] = -1
                        cpphigh[m+33] = -1
```

```
                #Randterme beruecksichtigen
                elif cpphigh[m] == 0 and cpplow[m]==0 and cpphigh[m-1] == -1:
                        cpphigh[m] = -1
                #Randterme auch in den letzten 33 Zeitintervallen beruecksichtigen
        for m in range(xmax-33, xmax):
                if cpphigh[m]==0 and cpphigh[m-1]==-1 and cpplow[m]==0:
                        cpphigh[m] = -1




        #Erstelle Histogrammdaten fuer Pixel i mit Klassen von 1-19 und einer Klasse von
20 bis 1000

        print 'Erstelle Histogrammdaten, Pixel {0}' .format(i)

        (f, bins) = np.histogram(cpphigh, bins=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 18, 19, 20, 1000, 100000))

        n.append(f)                     #Speichere Histogrammdaten fuer alle Pixel in
Liste n

##Oeffne Output-Datei
Hist = open(sys.argv[1], 'w')
# Speichere Histogrammdaten in Output-Datei, mit Klassenhoehen in einer Zeile und
Pixelnummer in letzter Spalte
for x in range(len(n)):
   print>>Hist  ,n[x][0],n[x][1],n[x][2],n[x][3],n[x][4],n[x][5],n[x][6],n[x][7],n[x][8],
n[x][9], n[x][10], n[x][11], n[x][12], n[x][13], n[x][14], n[x][15], n[x][16], n[x][17],
n[x][18], n[x][19], n[x][20], n[x][21], x+start



Hist.close()
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#============================================================================
import sys
import os
import string
import numpy as np
sys.path.append('/d6/alexgw/SOFTWARE/python-64bit') # 64bit!!!
import matplotlib
matplotlib.use('TkAgg')
#matplotlib.interactive(True)
import matplotlib.pyplot as plt
import math
import random
import scipy as sci
import scipy.integrate as integ
import pylab
#import astropysics.coords as C
#sys.path.append('/d6/alexgw/SOFTWARE/python-32bit') # 32bit!!!
import healpy
import matplotlib.cm as cm
import matplotlib.mlab as mlab
import matplotlib.pylab as mpl
import scipy.optimize as sco
import scipy.special as spec
import pyfits
import scipy.ndimage as ni
import matplotlib.pyplot as plt
from matplotlib import rc
import pylab
from pylab import *
import scipy

#============================================================================

#Input 1: Parameter Daten (1GeV)
#Input 2: Histogramm Daten

# Mit sys kann die zu oeffnende Datei als Argument eingegeben werden. So muss der Code
nicht fuer unterschiedliche Input-Dateien geaendert werden
#Oeffne Parameterdaten
Param = open(sys.argv[1])
#Nehme Werte aus der Parameter-Datei
naxis2=long(Param.readline())   # Anzahl an Ereignissen
healpix=long(Param.readline())  # Pixelanzahl

#Oeffne Histogrammdaten
ArrayInt = open(sys.argv[2])

#Lese Daten aus den Histogrammdaten des Python Programms GRB_Suche.py zeilenweise ein und
starte jedesmal erneut am Anfang der Datei

Z1 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(0))
ArrayInt.seek(0, 0)

Z2 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(1))
ArrayInt.seek(0, 0)

Z3 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(2))
```

```
ArrayInt.seek(0, 0)

Z4 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(3))
ArrayInt.seek(0, 0)

Z5 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(4))
ArrayInt.seek(0, 0)

Z6 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(5))
ArrayInt.seek(0, 0)

Z7 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(6))
ArrayInt.seek(0, 0)

Z8 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(7))
ArrayInt.seek(0, 0)

Z9 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(8))
ArrayInt.seek(0, 0)

Z10 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(9))
ArrayInt.seek(0, 0)

Z11 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(10))
ArrayInt.seek(0, 0)

Z12 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(11))
ArrayInt.seek(0, 0)

Z13 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(12))
ArrayInt.seek(0, 0)

Z14 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(13))
ArrayInt.seek(0, 0)

Z15 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(14))
ArrayInt.seek(0, 0)

Z16 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(15))
ArrayInt.seek(0, 0)

Z17 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(16))
ArrayInt.seek(0, 0)

Z18 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(17))
ArrayInt.seek(0, 0)

Z19 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(18))
ArrayInt.seek(0, 0)

Z20 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(19))
ArrayInt.seek(0, 0)

Z21 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(20))
ArrayInt.seek(0, 0)

ArrayInt.close()
```

```
#Erzeuge Liste mit so vielen Nullen wie der Anzahl an Pixeln. Eintraege für interessanten
Pixel werden spaeter veraendert
Interest = np.zeros(healpix)
#Erzeuge mit Nullen gefuellte Liste, in die die Parameter x_0 fuer jedes Pixel
hineingeschrieben werden
x0 = np.ones(healpix)


##############################################################################################
############################
#Erstelle Verteilung der inversen Rate x0

pix=len(Z1) # Anzahl an untersuchten Pixeln

for i in range(pix):
        if Z3[i]!=0:                                            # Fall 1: Klasse 2 enthält
Eintraege
                x = [0, 1, 2]                                   # Fitte Gerade an
logarithmierte Daten an; f(x)=Polycoeffs[1]*exp(-polycoeffs[0]*x)
                xdata = np.array(x)
                y = np.log([Z1[i], Z2[i], Z3[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)        #polyfit(x,y,Grad des
Polynoms) zum Fitten einer Polynomfunktion an die Daten
                x0[i]=-polycoeffs[0]                            #x0 entspricht negativer
Steigung der Geraden
        elif Z2[i]!=0:                                          # Fall 2: Klasse 1
enthaelt Eintraege, Klasse 2 nicht
                x = [0, 1]
                xdata = np.array(x)
                y = np.log([Z1[i], Z2[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)        #polyfit(x,y,Grad des
Polynoms) zum Fitten einer Polynomfunktion an die Daten
                x0[i]=-polycoeffs[0]
        else:                                                   #Fall 3: Weder Klasse 1,
noch Klasse 2 enthalten Eintraege, setze Wert gleich Null
                x0[i]=0

for i in range(pix, healpix):                                   # Nicht untersuchte Pixel
erhalten den Wert 6
        x0[i] = 6




print """
*******************************
*    visualize output file    *
*******************************"""

healpy.mollview(x0, fig = 1, coord = 'G', unit = '', title=r'Verteilung der inversen Rate
$\alpha$')
healpy.graticule(dmer = 10.0, dpar = 10.0, coord = 'G', color = 'black')

# ----------------------------
print "...plot figure"
```

```
#plt.savefig('/d4/ahirt/Jag/Histogramme/inverse_Rate.png')              #Speichere Karte
ab
plt.show()
plt.clf()



#Fitkurve durch die logarithmierten Messwerte fuer 0, 1 und 2 Events pro Zeitintervall und
Pixel
for i in range(pix):
        if Z2[i]!=0 and Z3[i]!=0 and Z4[i]!=0 and Z5[i] !=0 and Z6[i]!=0:       #Fuer
Eintraege in den Klassen 1,2,3,4,5
                x = [1, 2]
                xdata = np.array(x)
                y = np.log([Z2[i], Z3[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)                #numpy.polyfit(x,
y, deg, rcond=None, full=False, w=None, cov=False)
                if np.log(Z5[i])>= polycoeffs[0] * 4 + polycoeffs[1] or np.log(Z6[i])>=
polycoeffs[0] * 5 + polycoeffs[1]:

                        bins = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
                        counts = [np.log(Z1[i]+0.0001), np.log(Z2[i]+0.0001),
np.log(Z3[i]+0.0001), np.log(Z4[i]+0.0001), np.log(Z5[i]+0.0001), np.log(Z6[i]+0.0001),
np.log(Z7[i]+0.0001), np.log(Z8[i]+0.0001), np.log(Z9[i]+0.0001), np.log(Z10[i]+0.0001)]

                        pos = np.arange(len(bins))
                        width = 1.0     # Macht ein Histogramm aus dem Balkendiagramm

                        ax = plt.axes()
                        ax.set_xticks(pos + (width / 2))
                        ax.set_xticklabels(bins)
                        v=[0,7,-1,polycoeffs[0] * (-0.5) +
polycoeffs[1]+np.sqrt(polycoeffs[0] * (-0.5) + polycoeffs[1]) ]
                        plt.axis(v)
                        lx = plt.xlabel(r'$\frac{\# Ereignisse}{Pixel \cdot
Zeitintervall}$', fontsize=18)
#                       lx = plt.xlabel("Anzahl der Ereignisse pro Pixel und
Zeitintervall")
                        ly = plt.ylabel(r'$\log(\# Zeitintervalle)$', fontsize = 14)
#                       ly = plt.ylabel("log(Anzahl der Zeitintervalle)")
                        tl = plt.title("Pixelnummer {0}" .format(i))
                        a = np.round(x0[i], 3)
                        b = np.round(polycoeffs[1], 3)
                        plt.text(3.5, np.log(Z1[i])+1, r'$f(x)={}*exp(-{}*x)$' .format(b,
a))
                        #plt.text(3.5, np.log(Z1[i]), r'$\sigma = {}$' .format(err))
                        Interest[i]=1
                        plt.bar(pos, counts, width, color='b')          #Blaue
Histogrammbalken
                        errorbar(np.array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5]),
np.array(np.log([Z1[i], Z2[i], Z3[i], Z4[i], Z5[i], Z6[i]])),
yerr=np.log(sqrt(np.array([Z1[i], Z2[i], Z3[i], Z4[i], Z5[i], Z6[i]]))), fmt=None,
ecolor='r')
                        x = np.arange(0, -polycoeffs[1]/polycoeffs[0]+0.5, 0.01)
                        y = polycoeffs[0] * (x-0.5) + polycoeffs[1]     # x-0.5, wegen der
Achsenskalierung, sonst verschobener Graph
                        pylab.plot(x, y, 'r-')                          #rote Fitkurve
                        pylab.plot(xdata + 0.5, ydata, 'k.')            #schwarze
```

```
Datenpunkte
                        plt.show()
#                       plt.savefig('/d4/ahirt/Jag/Histogramme/Pixel_'+str(i)+'.png')
                        plt.clf()
                        print i, polycoeffs

        elif Z2[i]!=0 and Z3[i]!=0 and Z4[i]!=0 and Z5[i]!=0:
                x = [1, 2]
                xdata = np.array(x)
                y = np.log([Z2[i], Z3[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)                #numpy.polyfit(x,
y, deg, rcond=None, full=False, w=None, cov=False)
                if np.log(Z5[i])>= polycoeffs[0] * 4 + polycoeffs[1]:   #Bedingung für
interessante Pixel


                        bins = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
                        counts = [np.log(Z1[i]+0.0001), np.log(Z2[i]+0.0001),
np.log(Z3[i]+0.0001), np.log(Z4[i]+0.0001), np.log(Z5[i]+0.0001), np.log(Z6[i]+0.0001),
np.log(Z7[i]+0.0001), np.log(Z8[i]+0.0001), np.log(Z9[i]+0.0001),
np.log(Z10[i]+0.0001)]  #+0.0001, damit log gebildet werden kann

                        pos = np.arange(len(bins))
                        width = 1.0                                     # Macht ein
Histogramm aus dem Balkendiagramm

                        ax = plt.axes()
                        ax.set_xticks(pos + (width / 2))
                        ax.set_xticklabels(bins)
                        v=[0,7,-1,polycoeffs[0] * (-0.5) +
polycoeffs[1]+np.sqrt(polycoeffs[0] * (-0.5) + polycoeffs[1]) ]  #gibt Länge der Achsen an
                        plt.axis(v)
                        lx = plt.xlabel(r'$\frac{\# Ereignisse}{Pixel \cdot
Zeitintervall}$', fontsize=18)          #x-Achsenbeschriftung
                        ly = plt.ylabel(r'$\log(\# Zeitintervalle)$', fontsize = 14)
                                #y-Achsenbeschriftung
                        tl = plt.title("Pixelnummer {0}" .format(i))
                                #Titel
                        a = np.round(x0[i], 3)
                        b = np.round(polycoeffs[1], 3)
                        plt.text(3.5, np.log(Z1[i])+1, r'$f(x)={}*exp(-{}*x)$' .format(b,
a))                             #gibt exp-Funktion an
                        Interest[i]=1
                                #Ändert 0 --> 1 in Liste Interest
                        plt.bar(pos, counts, width, color='b')
                                #Blaue Histogrammbalken
                        errorbar(np.array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5]),
np.array(np.log([Z1[i], Z2[i], Z3[i], Z4[i], Z5[i], Z6[i]])),
yerr=sqrt(np.array(np.log([Z1[i], Z2[i], Z3[i], Z4[i], Z5[i], Z6[i]]))), fmt=None,
ecolor='r')
                        x = np.arange(0, -polycoeffs[1]/polycoeffs[0]+0.5, 0.01)
                        y = polycoeffs[0] * (x-0.5) + polycoeffs[1]                     #
x-0.5, wegen der Achsenskalierung, sonst verschobener Graph
                        pylab.plot(x, y, 'r-')
                                #rote Fitkurve
                        pylab.plot(xdata + 0.5, ydata, 'k.')
                                #schwarze Datenpunkte
                        plt.show()
```

```
#                       plt.savefig('/d4/ahirt/Jag/Histogramme/Pixel_'+str(i)+'.png')
                        plt.clf()
                        print i, polycoeffs


        elif Z6[i]!=0 or Z7[i]!=0 or Z8[i]!=0 or Z9[i]!=0 or Z10[i]!=0 or Z11[i]!=0 or
Z12[i]!=0 or Z13[i]!=0 or Z14[i]!=0 or Z15[i]!=0 or Z16[i]!=0 or Z17[i]!=0 or Z18[i]!=0 or
Z19[i]!=0 or Z20[i]!=0 or Z21[i]!=0:                    # Für transiente Ereignisse, nur
Hintergrund + Zeitintervall mit vielen Ereignissen
                x = [0, 1, 2]
                xdata = np.array(x)
                y = np.log([Z1[i], Z2[i], Z3[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)                #numpy.polyfit(x,
y, deg, rcond=None, full=False, w=None, cov=False)


                bins = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
                counts = [np.log(Z1[i]+0.0001), np.log(Z2[i]+0.0001),
np.log(Z3[i]+0.0001), np.log(Z4[i]+0.0001), np.log(Z5[i]+0.0001), np.log(Z6[i]+0.0001),
np.log(Z7[i]+0.0001), np.log(Z8[i]+0.0001), np.log(Z9[i]+0.0001), np.log(Z10[i]+0.0001)]

                pos = np.arange(len(bins))
                width = 1.0     # Macht ein Histogramm aus dem Balkendiagramm

                ax = plt.axes()
                ax.set_xticks(pos + (width / 2))
                ax.set_xticklabels(bins)
                lx = plt.xlabel(r'$\frac{\# Ereignisse}{Pixel \cdot Zeitintervall}$',
fontsize=18)
                ly = plt.ylabel(r'$\log(\# Zeitintervalle)$', fontsize = 14)
                tl = plt.title("Pixelnummer {0}" .format(i))
                a = np.round(x0[i], 3)
                b = np.round(polycoeffs[1], 3)
                plt.text(3.5, np.log(Z1[i])+1, r'$f(x)={}*exp(-{}*x)$' .format(b, a))
                Interest[i]=2                                                   #
Verändere 0 --> 2 in Interest, damit dieses Pixel vom Rest unterschieden werden kann
                plt.bar(pos, counts, width, color='b')                          #Blaue
Histogrammbalken
                errorbar(np.array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5]), np.array(np.log([Z1[i],
Z2[i], Z3[i], Z4[i], Z5[i], Z6[i]])), yerr=sqrt(np.array(np.log([Z1[i], Z2[i], Z3[i],
Z4[i], Z5[i], Z6[i]]))), fmt=None, ecolor='r')
                x = np.arange(0, -polycoeffs[1]/polycoeffs[0]+0.5, 0.01)
                y = polycoeffs[0] * (x-0.5) + polycoeffs[1]                     # x-0.5,
wegen der Achsenskalierung, sonst verschobener Graph
                pylab.plot(x, y, 'r-')                                          #rote
Fitkurve
                pylab.plot(xdata + 0.5, ydata, 'k.')                            #schwarze
Datenpunkte
                plt.show()
#               plt.savefig('/d4/ahirt/Jag/Histogramme/Pixel_'+str(i)+'.png')
                plt.clf()
                print i, polycoeffs


#*******************************************************************************************
*****************************************
```

```
#Plotte interessante Pixel in Himmelskarte
print """
*******************************
*    visualize output file    *
*******************************"""

healpy.mollview(Interest, fig = 1, coord = 'G', unit = '', title='Interessante Pixel')
healpy.graticule(dmer = 10.0, dpar = 10.0, coord = 'G', color = 'black')
# ----------------------------
print "...plot figure"

plt.show()
plt.clf()



Param.close()
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#============================================================================
import sys
import os
import string
import numpy as np
sys.path.append('/d6/alexgw/SOFTWARE/python-64bit') # 64bit!!!
import matplotlib
matplotlib.use('TkAgg')
#matplotlib.interactive(True)
import matplotlib.pyplot as plt
import math
import random
import scipy as sci
import scipy.integrate as integ
import pylab
#import astropysics.coords as C
#sys.path.append('/d6/alexgw/SOFTWARE/python-32bit') # 32bit!!!
import healpy
import matplotlib.cm as cm
import matplotlib.mlab as mlab
import matplotlib.pylab as mpl
import scipy.optimize as sco
import scipy.special as spec
import pyfits
import scipy.ndimage as ni
import matplotlib.pyplot as plt
from matplotlib import rc
import pylab
from pylab import *
import scipy

#============================================================================

#Input: Histogramm Daten

# Mit sys kann die zu oeffnende Datei als Argument eingegeben werden. So muss der Code
nicht fuer unterschiedliche Input-Dateien geaendert werden
#Oeffne Histogrammdaten
ArrayInt = open(sys.argv[1])

#Lese Daten aus den Histogrammdaten des ersten Python Programms zeilenweise ein und starte
jedesmal erneut am Beginn der Datei

Z1 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(0))
ArrayInt.seek(0, 0)

Z2 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(1))
ArrayInt.seek(0, 0)

Z3 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(2))
ArrayInt.seek(0, 0)

Z4 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(3))
ArrayInt.seek(0, 0)

Z5 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(4))
ArrayInt.seek(0, 0)
```

```
Z6 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(5))
ArrayInt.seek(0, 0)

Z7 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(6))
ArrayInt.seek(0, 0)

Z8 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(7))
ArrayInt.seek(0, 0)

Z9 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(8))
ArrayInt.seek(0, 0)

Z10 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(9))
ArrayInt.seek(0, 0)

Z11 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(10))
ArrayInt.seek(0, 0)

Z12 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(11))
ArrayInt.seek(0, 0)

Z13 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(12))
ArrayInt.seek(0, 0)

Z14 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(13))
ArrayInt.seek(0, 0)

Z15 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(14))
ArrayInt.seek(0, 0)

Z16 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(15))
ArrayInt.seek(0, 0)

Z17 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(16))
ArrayInt.seek(0, 0)

Z18 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(17))
ArrayInt.seek(0, 0)

Z19 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(18))
ArrayInt.seek(0, 0)

Z20 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(19))
ArrayInt.seek(0, 0)

Z21 = np.genfromtxt(ArrayInt, delimiter=' ', dtype=int, usecols=(20))
ArrayInt.seek(0, 0)

#print 'schliesse Datei'
ArrayInt.close()



#Erzeuge mit Nullen gefuelltes array, in das die der maximale Eintrag fuer jedes Pixel
hineingeschrieben wird
nmax = np.zeros(len(Z1))
#Suche nach maximalem Eintrag
```

```
for i in range(len(Z1)):
        if Z21[i]!=0:
                nmax[i]=20
        elif Z20[i]!=0:
                nmax[i]=19
        elif Z19[i]!=0:
                nmax[i]=18
        elif Z18[i]!=0:
                nmax[i]=17
        elif Z17[i]!=0:
                nmax[i]=16
        elif Z16[i]!=0:
                nmax[i]=15
        elif Z15[i]!=0:
                nmax[i]=14
        elif Z14[i]!=0:
                nmax[i]=13
        elif Z13[i]!=0:
                nmax[i]=12
        elif Z12[i]!=0:
                nmax[i]=11
        elif Z11[i]!=0:
                nmax[i]=10
        elif Z10[i]!=0:
                nmax[i]=9
        elif Z9[i]!=0:
                nmax[i]=8
        elif Z8[i]!=0:
                nmax[i]=7
        elif Z7[i]!=0:
                nmax[i]=6
        elif Z6[i]!=0:
                nmax[i]=5
        elif Z5[i]!=0:
                nmax[i]=4
        elif Z4[i]!=0:
                nmax[i]=3
        elif Z3[i]!=0:
                nmax[i]=2
        elif Z2[i]!=0:
                nmax[i]=1
        elif Z1[i]!=0:
                nmax[i]=0


ngamma=np.zeros(len(Z1))
lamb=np.zeros(len(Z1))

for i in range(len(Z1)):
        if Z2[i]!=0 and Z3[i]==0:               # wenn Klasse 1 Eintraege hat und Klasse 2
nicht
                x = [0, 1]
                xdata = np.array(x)
                y = np.log([Z1[i], Z2[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)
                erwartung= polycoeffs[1]*np.exp(polycoeffs[0] * nmax[i])        # Bilde
Funktionswert bei nmax
                print erwartung
```

```
                lamb[i]=erwartung
                n= 1/polycoeffs[0]*np.log(0.00001/polycoeffs[1])                #Bilde
n(0,00001)
                print n
                ngamma[i]=n                                                     # Schreibe
n(0,00001) an die i-te Stelle der Liste


        elif Z2[i]==0 and Z3[i]!=0:             # wenn Klasse 2 Eintraege hat und Klasse 1
nicht
                x = [0, 2]
                xdata = np.array(x)
                y = np.log([Z1[i], Z3[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)
                bound=-polycoeffs[1]/polycoeffs[0]
                erwartung= polycoeffs[1]*np.exp(polycoeffs[0] * nmax[i])
                print erwartung
                lamb[i]=erwartung
                n= 1/polycoeffs[0]*np.log(0.00001/polycoeffs[1])
                print n
                ngamma[i]=n


        elif Z2[i]!=0 and Z3[i]!=0:             # wenn Klassen 1 und 2 Eintraege haben
                x = [0, 1, 2]
                xdata = np.array(x)
                y = np.log([Z1[i], Z2[i], Z3[i]])
                ydata = np.array(y)
                polycoeffs = np.polyfit(xdata, ydata, 1)
                bound=-polycoeffs[1]/polycoeffs[0]
                erwartung= polycoeffs[1]*np.exp(polycoeffs[0] * nmax[i])
                print erwartung
                lamb[i]=erwartung
                n= 1/polycoeffs[0]*np.log(0.00001/polycoeffs[1])
                print n
                ngamma[i]=n



summe=sum(ngamma)
mittel=summe/len(Z1)                            #Bilde den Mittelwert der n(0,00001) jedes
Pixels
print 'Mittelwert von n ist {0}.' .format(mittel)
mittlamb= sum(lamb)/len(Z1)                     # Bilde den Mittelwert von lambda
print 'Mittelwert von lambda ist {0}.' .format(mittlamb)
nmaxi=sum(nmax)/len(Z1)                         # Mittelwert der maximal in einem
Zeitintervall aufgetretenen Ereignisse
print 'Mittelwert nmax={0}' .format(nmaxi)
```

Literaturverzeichnis

[Atwood et al. 2009] Atwood et al.: The Large Area Telescope on the Fermi Gamma-Ray Space Telescope mission. In: *The Astrophysical Journal* (2009). – arXiv:0902.1089

[Bains 2011] Bains, Jagdev: *Searching for Evaporating Primordial Black Holes using Fermi Gamma Ray Telescope Data*, Universität Hamburg, Diplomarbeit, 2011

[Bouvier 2010] Bouvier, Aurélien P.: *Gamma-ray Bursts observations at high-energy with the Fermi Large Area Telescope*, Stanford University, Dissertation, 2010

[C.A. Swenson et al. 2010] C.A. Swenson, A. M. et al.: *GRB 090926A and bright late-time Fermi LAT GRB afterglows*. 2010. – arXiv: 1004.5099v1

[CDS] CDS: *SIMBAD Astronomical Database*. – URL http://simbad.u-strasbg.fr/simbad/. – Zugriffsdatum: 30.03.2012

[F. Fürst et al. 2009] F. Fürst, R.E. R. et al.: Temporal Variations of Strength and Location of the South Atlantic Anomaly as Measured by RXTE. In: *Earth and Planetary Science Letters* (2009). – arXiv: 0902.2873v1

[Fermi Science Support Center a] Fermi Science Support Center: *Cicerone: Data - LAT Data Products*. – URL http://fermi.gsfc.nasa.gov/ssc/data/analysis/documentation/Cicerone/Cicerone_Data/LAT_DP.html. – Zugriffsdatum: 10.03.2012

[Fermi Science Support Center b] Fermi Science Support Center: *Data Preparation*. – URL http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/data_preparation.html. – Zugriffsdatum: 30.03.2012

[Fermi Science Support Center c] Fermi Science Support Center: *Installing the Fermi Science Tools*. – URL http://fermi.gsfc.nasa.gov/ssc/data/analysis/software/

[Fermi Science Support Center d] Fermi Science Support Center: *Orbit Simulation*. – URL http://fermi.gsfc.nasa.gov/ssc/data/analysis/documentation/Cicerone/Cicerone_Obs_Sim/orbit_simulation.html. – Zugriffsdatum: 30.03.2012

[Fermi Science Support Center 2012] Fermi Science Support Center: *Caveats About Analyzing LAT Pass 7 Data*. 2012. – URL http://fermi.gsfc.nasa.gov/ssc/data/analysis/LAT_caveats.html. – Zugriffsdatum: 30.03.2012

[G. D. Kribs 1999] G. D. Kribs, I. Z. R.: *Bounds from Primordial Black Holes with a Near Critical Collapse Initial Mass Function*. 1999. – arXiv: astro-ph/9904021v2

[Górski et al. 2005] Górski, K.M. et al.: HEALPix: A framework for high-resolution discretization and fast analysis. In: *The Astrophysical Journal* (2005). – URL http://iopscience.iop.org/0004-637X/622/2/759/pdf/61342.web.pdf. – Zugriffsdatum: 30.03.2012

[Górski] Górski, Krzysztof M.: *Healpix*. – URL http://healpix.jpl.nasa.gov/. – Zugriffsdatum: 24.02.2012

[Górski 2010] Górski, Krzysztof M.: *The Healpix Primer*. 2010. – URL http://healpix.jpl.nasa.gov/pdf/intro.pdf. – Zugriffsdatum: 24.02.2012

[Harvard-Smithsonian Center for Astrophysics 2008] Harvard-Smithsonian Center for Astrophysics: Supermassive Black Holes. (2008). – URL http://chandra.harvard.edu/xray_sources/blackholes_sm.html. – Zugriffsdatum: 30.03.2012

[HEASARC 2011] HEASARC: *The FITS Support Office*. 2011

[Hivon et al. 2010] Hivon, Eric et al.: *HEALPix C Subroutines Overview*. http://healpix.jpl.nasa.gov/pdf/csub.pdf: , 2010

['t Hooft 2009] Hooft, G. 't: *Introduction to the theory of black holes*. 2009

[J.M. Overduin 2004] J.M. Overduin, P.S. W.: *Dark Matter and Background Light*. 2004. – arXiv: astro-ph/0407207v1

[Keane 2011] Keane, E.F.: *The Transient Radio Sky*. Springer Verlag, 2011

[Michael S. Briggs 2000] Michael S. Briggs: *The BATSE Current Gamma-Ray Burst Catalog*. 2000. – URL http://www.batse.msfc.nasa.gov/batse/grb/skymap/. – Zugriffsdatum: 10.03.2012

[Mitchell et al.] Mitchell et al.: *Gamma-Ray Bursts: Introduction to a Mystery*. – URL http://imagine.gsfc.nasa.gov/docs/science/know_l1/grbs.html. – Zugriffsdatum: 30.03.2012

[Müller 2007] Müller, Andreas: *Lexikon der Astrophysik*. 2007. – URL http://www.wissenschaft-online.de/astrowissen/downloads/Lexikon/Lexikon_AMueller2007.pdf. – Zugriffsdatum: 27.02.2012

[NASA a] NASA: http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/help/gtselect.txt:

[NASA b] NASA: http://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/help/gtmktime.txt:

[NASA c] NASA: *Cicerone: Data - LAT Data Files - Column Descriptions.* – URL http://fermi.gsfc.nasa.gov/ssc/data/analysis/documentation/Cicerone/Cicerone_Data/LAT_Data_Columns.html

[NASA d] NASA: *Official NASA-Homepage.* – URL http://www.nasa.gov/. – Zugriffsdatum: 30.03.2012

[P. Mézáros 2002] P. Mézáros, B. Z.: An Analysis of Gamma-Ray Burst Spectral Break Models. In: *The Astrophysical Journal* (2002)

[R. Bousso 1996] R. Bousso, S.W. Hawking: *Primordial Black Holes: Tunnelling vs. No Boundary Proposal.* 1996. – arXiv: gr-qc/9608009v1

[S. W. Hawking 1994] S. W. Hawking: *The Nature of Space and Time.* 1994. – arxiv:hep-th/9409195

[Schutz 2004] Schutz, Bernhard: *Gravity from the Ground Up: An Introductory Guide to Gravity and General Relativity.* Cambridge University Press, 2004. – ISBN 978-0521455060

[S.E. Woosley 2003] S.E. Woosley, A.Heger: The Collapsar Model for Gamma-Ray Bursts. In: *Los Alamos NATIONAL LABORATORY* (2003). – URL http://library.lanl.gov/cgi-bin/getfile?LA-UR-03-9090sc.pdf. – Zugriffsdatum: 30.03.2012

[Y. Sofue 2008] Y. Sofue, T. O.: *Unified Rotation Curve of the Galaxy - Decomposition into de Vaucouleurs Bulge, Disk, Dark Halo, and the 9-kpc Rotation Dip -.* 2008. – arXiv: 0811.0859v2

Erklärung:

Hiermit bestätige ich, Alicia Hirt, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe und die Arbeit von mir vorher nicht einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium. Ich bin damit einverstanden, dass die Bachelorarbeit veröffentlicht wird.

Hamburg, den 03.04.2012

Alicia Hirt