References

Atkinson, E. K. An introduction to numerical analysis. 2nd ed. John Wiley & Sons, 1989.

Bieber, J. W. et al. Energetic particle observations during the 2000 July 14 solar event. Astrophys. J. **567** (2002): 622-634.

Bieber, J. W. et al. Proton and electron mean free paths: The palmer consensus revisited. Astrophys. J. **420** (1994): 294.

Cane, H. V. and Stone, R. G. Type II solar radio bursts, interplanetary shocks and energetic particle events. Astrophys. J. **282** (1984): 339-344.

Cliver, E. W. et al. Solar flare nuclear gamma-rays and interplanetary proton events. Astrophys. J. **343** (1989): 953-970.

Cliver, E. W. et al. Particle injection in the 6 November 1997 SEP event: CME, radio and gamma-ray observations. Proc. Intl. Cosmic Ray Conf. (2001): 3277.

Cohen, C. M. S. et al. Inferred charge state of high energy solar particles from the Solar Isotope Spectrometer on ACE. Geophys. Res. Lett. **26** (1999): 149.

Earl, J. A. Diffusion on charged particles in a random magnetic field. Astrophys. J. **180** (1973): 227.

Earl, J. A. The effect of adiabatic focusing upon charged-particle propagation in random magnetic fields. Astrophys. J. **205** (1976): 900.

Harten, A. High resolution schemes for hyperbolic conservation laws. J. Comput. Phys. **49** (1983): 357.

Hasselmann, K., and Wibberenz, G. Scattering of particles by random electromagnetic field. Z. Geophys. **34** (1968): 353.

Jokipii, J. R. Cosmic-ray propagation. I. Charged particles in a random magnetic field. Astrophys. J. **146** (1966): 480; Addendum and erratum to "Cosmic-ray propagation. I", Astrophys. J. **152** (1968): 671.

Jokipii, J. R. Propagation of cosmic rays in the solar wind. Rev. Geophys. Space Phys. **9** (1971): 27.

Klein, K. and Trottet, G. Post-impulsive coronal particle acceleration and its possible role in the large SEP event of 6 November 1997: Radio evidence. American Geophysical Union (2001).

Klein, K.-L. et al. Coronal electron acceleration and relativistic proton production during the 14 July 2000 flare and CME. Astronomy and Astrophysics **373** (2001): 1073-1082.

Laitinen, T. et al. Solar energetic particle event and radio bursts associated with the 1996 July 9 flare and coronal mass ejection. Astron. Astrophys. **360** (2000): 729

Leske, R. A. Unusual isotopic composition of solar energetic particles observed in the November 6, 1997 event. Geophys. Res. Lett. **26** (1999): 153.

Mason, G. M. et al. Particle acceleration and sources in the November 1997 solar energetic particle events. Geophys. Res. Lett. **26** (1999a): 141.

Mason, G. M. et al. ^{3}He enhancements in large solar energetic particle events. Astrophys. J. **525** (1999b): L133-L136.

Mathews, H. J. and Fink, D. K. Numerical methods using mathlab. 3rd ed. Prentice Hall International, Inc., 1999.

Möbius, E. et al. Energy dependence of the ionic charge state distribution during the November 1997 solar energetic particle event. Geophys. Res. Lett. **26** (1999): 145.

Nutaro, T., Riyavong, S. and Ruffolo, D. Application of a generalized total variation diminishing algorithm to cosmic ray transport and acceleration. Computer Physics Communications. **134** (2001): 209-222.

Pallavicini, R. et al. A survey of soft X-ray limb flare images - The relation between their structure in the corona and other physical parameters. Astrophys. J. **216** (1997): 108-122.

Pinsky, A. M. Partial differential equation and boundary-value problems with application. 3rd ed. The McGraw-Hill Companies, Inc., 1998.

Reames, D. V. et al. Heavy ion abundances and spectra and the large gradual solar energetic particle event of 2000 July 14. Astrophys. J. **548** (2001): L233-L236.

Roe, P. L. Some contributions to the modelling of discontinuous flows, in: B.E. Engquist et al. (Eds.), Proc. AMS/SIAM Sum. Sem. on Large-Scale Comp. in Fluid Mech., 1983, Lectures in Appl. Math., **22 (2)**, Amer. Math. Soc., Providence, RI, 1985

Roelof, E. C. Propagation of solar cosmic rays in the interplanetary magnetic field, in Lectures in High Energy Astrophysics ed. H. Ogelman & J. R. Wayland (NASA SP-199) (Washington: NASA) (1969), 111-135.

Ruffolo, D. Effect of adiabatic deceleration on the focused transport of solar cosmic rays. Astrophys. J. **442** (1995), 861-874.

Ruffolo, D.,and Khumlumlert, T. Formation, propagation, and decay of coherent pulses of solar cosmic rays. Geophys. Res. Lett., **22** (1995): 2073-2076.

Ruffolo, D. Khumlumlert, T. and Youngdee, W. Deconvolution of interplanetary transport of solar energetic particles. J. Geophys. Res., **103** (1998): 20591.

Share, G. H. et al. Gamma-ray line observations of the 2000 July 14 flare and SEP impact on the Earth. Proc. of Intl. Cosmic Ray Conf. (2001).

Tylka, A. J. et al. Evidence for remnant flare suprathermals in the Source population of solar energetic particles in the 2000 Bastille Day event. Astrophys. J. **558** (2001): L59-L63.

Yan, Y. et al. Evolution of magnetic flux rope in the active region NOAA 9077 on 14 July 2000. Solar Physics. **204** (2001): 27-40.

Appendices

Appendix A
Uncertainty of Interplanetary Fluctuations

An example of our new technique for estimating the uncertainty of the spacecraft data, taking interplanetary fluctuations (IPF) in to account is shown in following figures. The ideas underlying this procedure are in section 4.3. The first column is the time of detecting particles in minutes, the second column is the intensity of the particles, and the third column contains counts of particles. The fourth column is σ_{stat}, which is calculated from the particle intensity divided by the square root of the particle counts. The fifth column is the particle intensity according to the trend line of the data of interest. The sixth column is the square of the difference between the second column and the fifth column. The seventh column is the ratio between the square of the intensity and the square of the average of the data of interest. The eighth column is an estimate of the IPF uncertainty, and the last column is the combined uncertainty (statistical + IPF) that we use.

time(min)	flux	count_flux	sig_stat	y(x_l)	[y-y_l]^2	I^2/I_0^2	sig_ipf	sig_tot
30	9.3E-07	1	9.3E-07	0.0005269	2.767E-07	2.106E-06	1.709E-07	9.456E-07
90	8.75E-07	1	8.75E-07	0.000532	2.821E-07	1.864E-06	1.608E-07	8.896E-07
150	0	0	#DIV/0!	0.0005371	2.884E-07	0	0	#DIV/0!
210	0	0	#DIV/0!	0.0005421	2.939E-07	0	0	#DIV/0!
270	0	0	#DIV/0!	0.0005472	2.994E-07	0	0	#DIV/0!
330	1.84E-06	2	1.301E-06	0.0005523	3.03E-07	8.242E-06	3.381E-07	1.344E-06
390	8.671E-07	1	8.671E-07	0.0005574	3.097E-07	1.831E-06	1.593E-07	8.817E-07
450	0	0	#DIV/0!	0.0005624	3.163E-07	0	0	#DIV/0!
510	9.171E-07	1	9.171E-07	0.0005675	3.21E-07	2.048E-06	1.685E-07	9.325E-07
570	8.04E-07	1	8.04E-07	0.0005726	3.269E-07	1.574E-06	1.477E-07	8.175E-07
630	0	0	#DIV/0!	0.0005776	3.337E-07	0	0	#DIV/0!
690	8.664E-07	1	8.664E-07	0.0005827	3.385E-07	1.828E-06	1.592E-07	8.809E-07
750	0	0	#DIV/0!	0.0005878	3.455E-07	0	0	#DIV/0!
810	7.412E-06	4	3.706E-06	0.0005928	3.427E-07	0.0001338	1.362E-06	3.948E-06
870	0.0001023	39	1.638E-05	0.0005979	2.457E-07	0.0254685	1.879E-05	2.493E-05
930	0.0002246	71	2.666E-05	0.000603	1.431E-07	0.1228559	4.127E-05	4.914E-05
990	0.0004124	127	3.659E-05	0.0006081	3.829E-08	0.4139957	7.577E-05	8.414E-05
1050	0.0006045	193	4.352E-05	0.0006131	7.37E-11	0.8897578	0.0001111	0.0001193
1110	0.0006144	204	4.302E-05	0.0006182	1.455E-11	0.9189584	0.0001129	0.0001208
1170	0.0006584	220	4.439E-05	0.0006233	1.237E-09	1.05549	0.000121	0.0001289
1230	0.000778	260	4.825E-05	0.0006283	2.24E-08	1.4736423	0.000143	0.0001509
1290	0.0004644	195	3.325E-05	0.0006334	2.857E-08	0.5250114	8.532E-05	9.158E-05
1350	0.0006607	239	4.274E-05	0.0006385	4.957E-10	1.0628768	0.0001214	0.0001287
1410	0.0005547	198	3.942E-05	0.0006435	7.893E-09	0.749097	0.0001019	0.0001093
1470	0.0006856	208	4.754E-05	0.0006486	1.369E-09	1.1443951	0.000126	0.0001346
1530	0.000702	213	4.81E-05	0.0006537	2.33E-09	1.1995934	0.000129	0.0001377
1590	0.0004281	183	3.165E-05	0.0006588	5.321E-08	0.4461613	7.866E-05	8.478E-05
1650	0.0004166	181	3.096E-05	0.0006638	6.114E-08	0.4224515	7.654E-05	8.256E-05
1710	0.0005537	207	3.849E-05	0.0006689	1.326E-08	0.7465064	0.0001017	0.0001088
1770	0.0004748	187	3.472E-05	0.000674	3.966E-08	0.5488829	8.724E-05	9.39E-05
1830	0.0003704	155	2.975E-05	0.000679	9.525E-08	0.3340313	6.806E-05	7.428E-05
1890	0.0003107	142	2.607E-05	0.0006841	1.394E-07	0.2349894	5.708E-05	6.276E-05
1950	0.0002743	137	2.344E-05	0.0006892	1.721E-07	0.1832314	5.041E-05	5.559E-05

Table A.1: Table of uncertainty estimation. Times 1230-1530 min represent the data of interest, used to estimate the IPF uncertainty.

time(min)	flux	count_flux	sig_stat	y(x_I)	[y-y_I]^2	I^2/I_0^2	sig_ipf	sig_tot
2010	0.000272	146	2.251E-05	0.0006942	1.783E-07	0.1801587	4.998E-05	5.482E-05
2070	0.0001936	116	1.798E-05	0.0006993	2.557E-07	0.0912782	3.558E-05	3.986E-05
2130	0.0001971	118	1.815E-05	0.0007044	2.573E-07	0.0945887	3.622E-05	4.051E-05
2190	0.000164	103	1.616E-05	0.0007095	2.975E-07	0.0654721	3.013E-05	3.419E-05
2250	0.0001712	113	1.611E-05	0.0007145	2.952E-07	0.0713558	3.146E-05	3.534E-05
2310	0.0001522	104	1.492E-05	0.0007196	3.22E-07	0.0563815	2.796E-05	3.169E-05
2370	0.0001247	86	1.345E-05	0.0007247	3.6E-07	0.0378517	2.291E-05	2.656E-05
2430	0.0001501	110	1.431E-05	0.0007297	3.36E-07	0.0548581	2.758E-05	3.107E-05
2490	0.0001418	117	1.311E-05	0.0007348	3.517E-07	0.0489317	2.605E-05	2.916E-05
2550	9.514E-05	75	1.099E-05	0.0007399	4.157E-07	0.0220377	1.748E-05	2.065E-05
2610	7.317E-05	58	9.608E-06	0.0007449	4.513E-07	0.0130343	1.344E-05	1.652E-05
2670	8.077E-05	65	1.002E-05	0.00075	4.479E-07	0.015881	1.484E-05	1.79E-05
2730	6.97E-05	57	9.231E-06	0.0007551	4.698E-07	0.011826	1.281E-05	1.579E-05
2790	4.615E-05	38	7.486E-06	0.0007602	5.098E-07	0.0051843	8.479E-06	1.131E-05
2850	7.019E-05	57	9.296E-06	0.0007652	4.831E-07	0.0119925	1.29E-05	1.59E-05
2910	8.437E-05	70	1.008E-05	0.0007703	4.705E-07	0.0173304	1.55E-05	1.849E-05
2970	6.297E-05	54	8.569E-06	0.0007754	5.075E-07	0.0096539	1.157E-05	1.44E-05
3030	5.654E-05	48	8.16E-06	0.0007804	5.24E-07	0.0077817	1.039E-05	1.321E-05
3090	3.666E-05	32	6.48E-06	0.0007855	5.608E-07	0.0032718	6.736E-06	9.347E-06
3150	3.503E-05	31	6.292E-06	0.0007906	5.708E-07	0.0029875	6.436E-06	9.001E-06
3210	4.385E-05	39	7.021E-06	0.0007956	5.652E-07	0.0046804	8.056E-06	1.069E-05
3270	2.859E-05	26	5.606E-06	0.0008007	5.962E-07	0.0019894	5.252E-06	7.682E-06
3330	2.819E-05	26	5.529E-06	0.0008058	6.047E-07	0.0019348	5.18E-06	7.576E-06
3390	4.543E-05	42	7.01E-06	0.0008109	5.859E-07	0.0050242	8.347E-06	1.09E-05
3450	3.639E-05	37	5.982E-06	0.0008159	6.077E-07	0.0032234	6.686E-06	8.971E-06
3510	2.244E-05	21	4.896E-06	0.000821	6.377E-07	0.0012255	4.122E-06	6.4E-06
3570	2.488E-05	24	5.078E-06	0.0008261	6.419E-07	0.0015064	4.57E-06	6.832E-06
3630	1.656E-05	16	4.14E-06	0.0008311	6.635E-07	0.0006677	3.043E-06	5.138E-06
3690	1.733E-05	17	4.204E-06	0.0008362	6.706E-07	0.0007313	3.185E-06	5.274E-06
3750	2.205E-05	22	4.701E-06	0.0008413	6.711E-07	0.0011835	4.051E-06	6.205E-06
3810	1.733E-05	17	4.203E-06	0.0008463	6.873E-07	0.0007313	3.184E-06	5.273E-06
3870	1.097E-05	11	3.308E-06	0.0008514	7.063E-07	0.000293	2.016E-06	3.874E-06

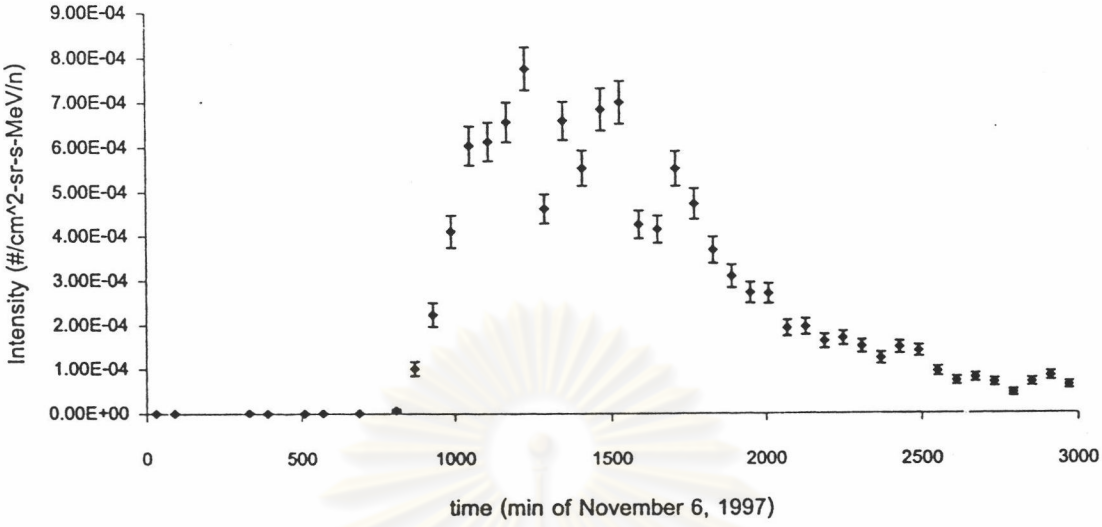


Figure A.1: Sample of the intensity of particles with σ_{stat} only. Note that the actual fluctuations are much greater than the error bars.

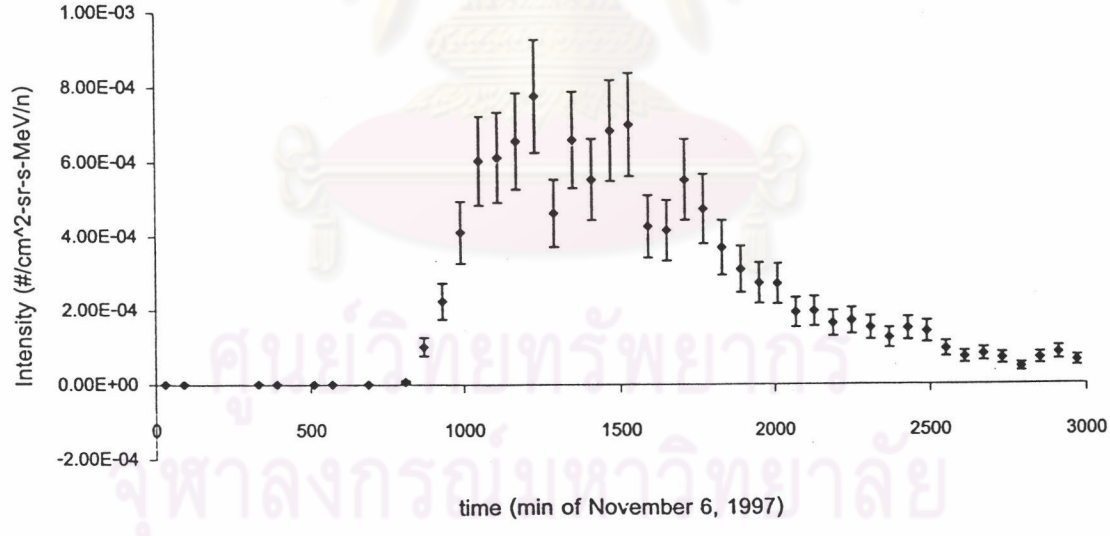


Figure A.2: Sample of the intensity of particles with combined uncertainties ($\sqrt{\sigma_{stat}^2 + \sigma_{IPF}^2}$). Now the error bars better represent the actual fluctuations.

Appendix B
The Minimum χ^2 Point

Lambda	chi-square
0.01	64.21
0.015	60.33
0.02	55.28
0.03	31.69
0.04	29.89
0.06	29.26
0.08	31.5
0.1	36.89
0.12	45.52
0.15	63.56
0.2	102.19
0.3	188.73
0.4	269.88
0.5	340.28
0.6	399.9
0.7	449.88
0.8	491.57
0.9	526.28
1	555.13
1.2	599.14
1.5	640.94
2	675.09
2.5	688.77
3	694.33

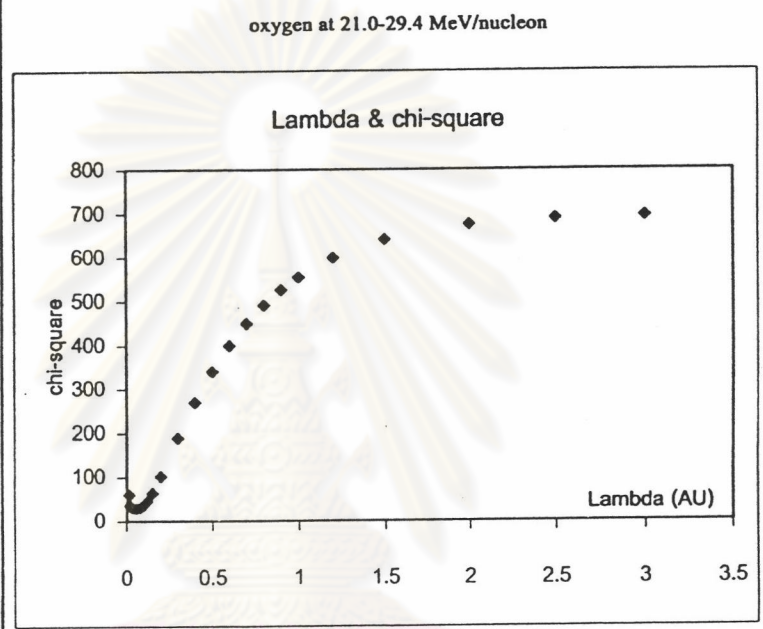
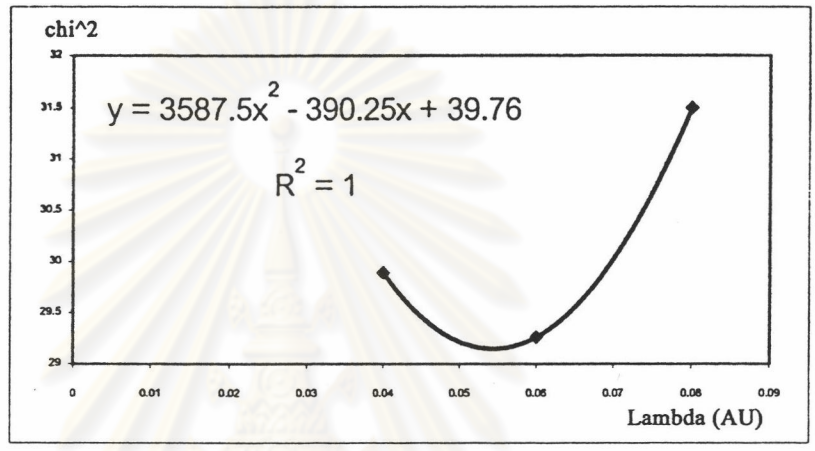


Figure B.1: Example of the χ^2 value at various λ.

find lambda_min & Lambda_1 and Lambda_2

Lambda	chi-square
0.04	29.89
0.06	29.26
0.08	31.5



x=(390.25)/(3587.5*2)

Lambda_min	0.05439
chi-square at L_min	29.1474
chi-square at L_min+1	30.1474
find L_1 and L_2	0.071088 L_1
	0.037692 L_2
delta_L	0.016698
	-0.016698

Figure B.2: Example of the minimum point from a parabolic graph.

Appendix C
Fitting Results for
November 6, 1997

These are the fitting results for the solar event on November 6, 1997.

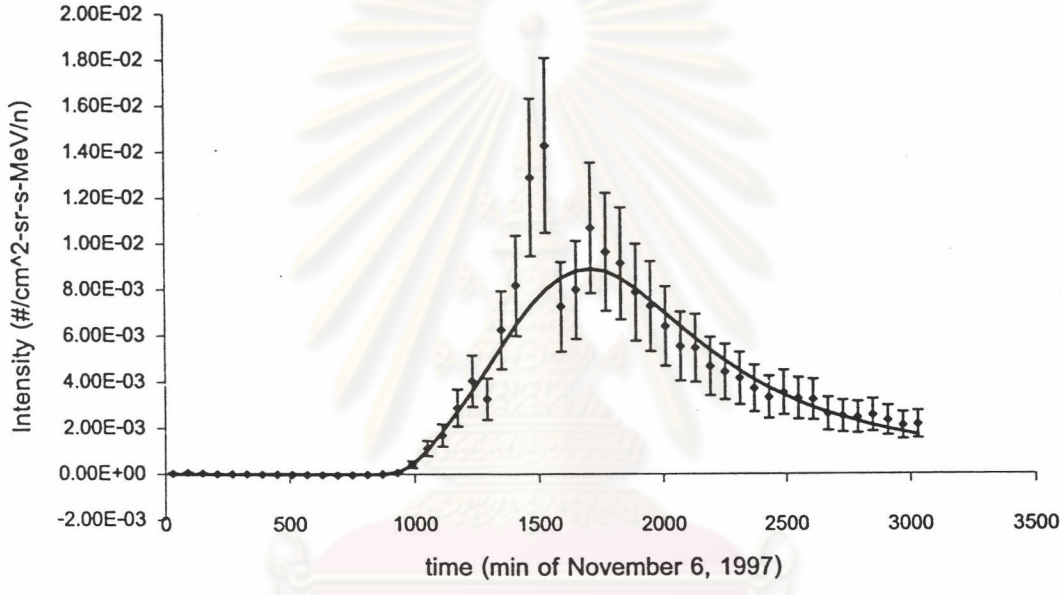


Figure C.1: The intensity fitting results of oxygen at 7.1-10.0 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

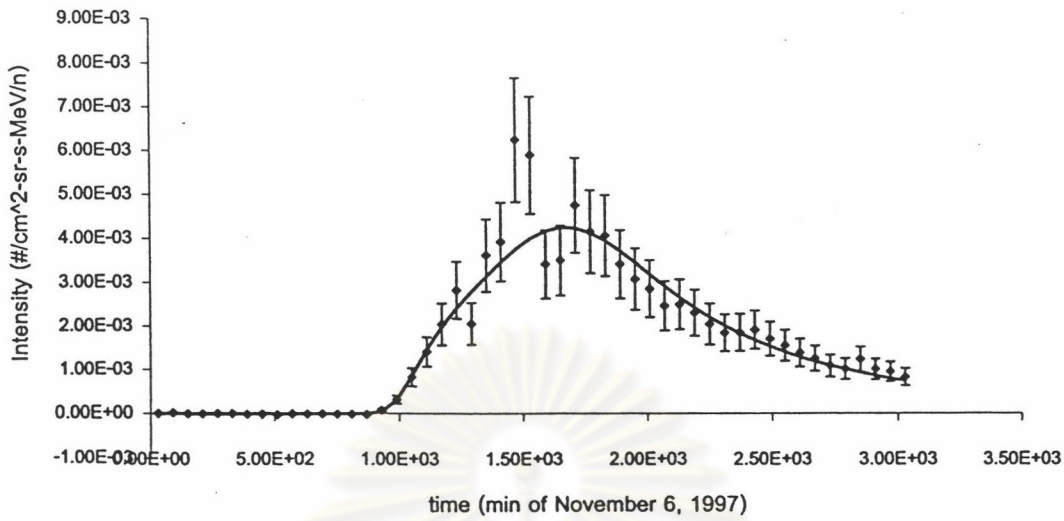


Figure C.2: The intensity fitting results of oxygen at 10.0-13.1 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

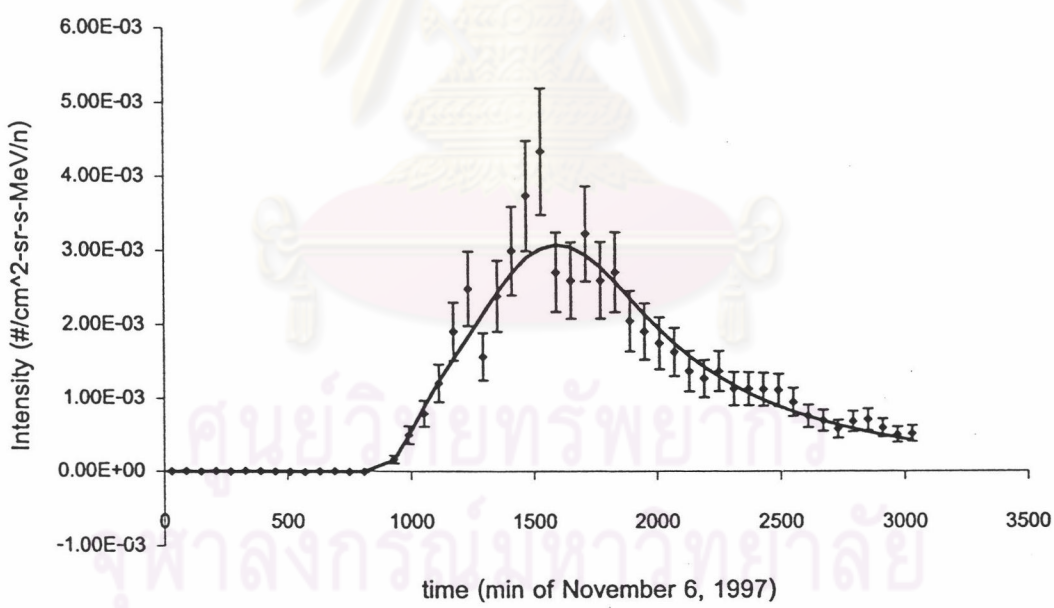


Figure C.3: The intensity fitting results of oxygen at 13.1-15.6 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

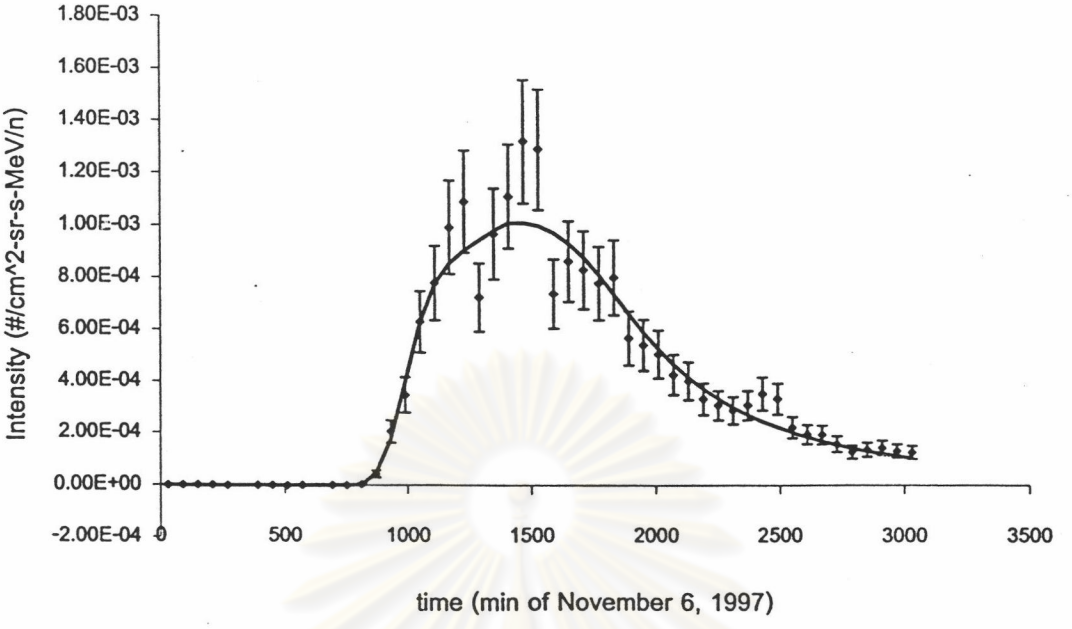


Figure C.4: The intensity fitting results of oxygen at 21.0-29.4 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

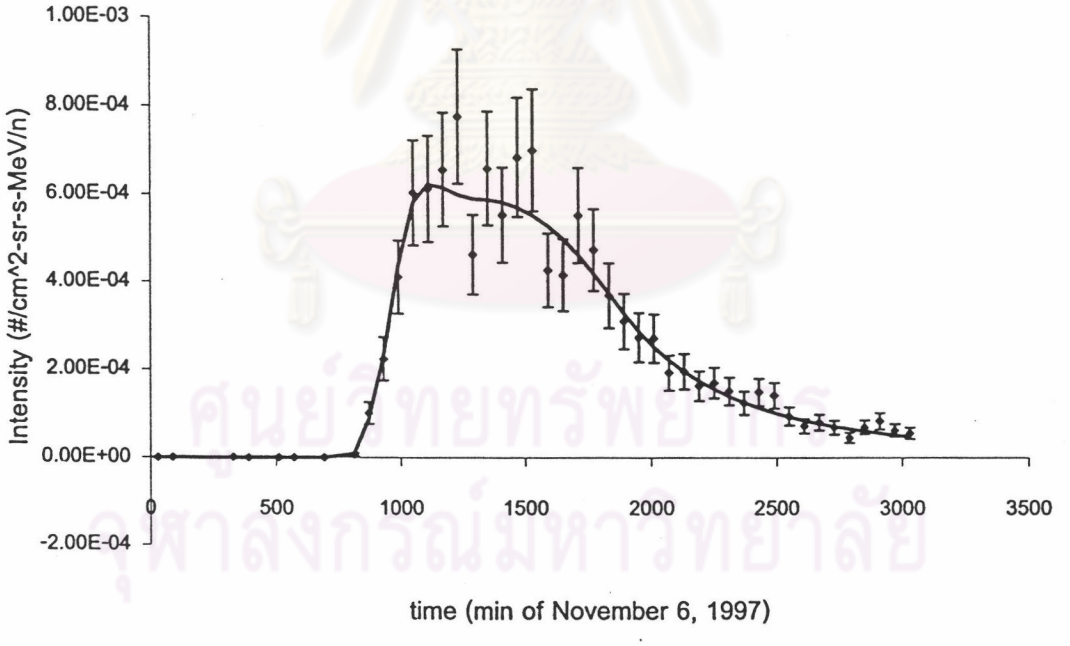


Figure C.5: The intensity fitting results of oxygen at 29.4-38.9 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

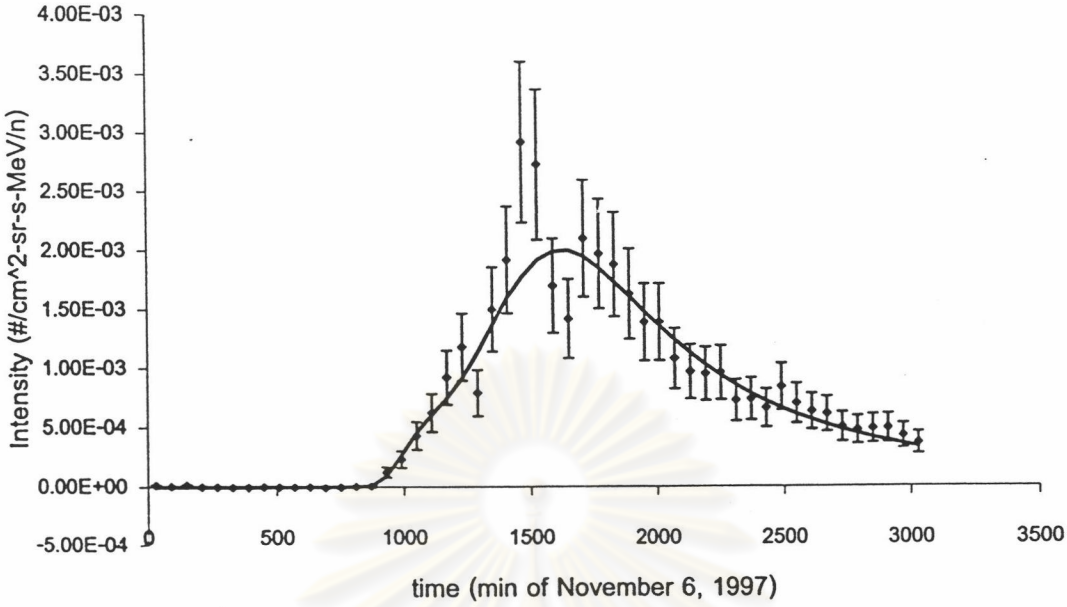


Figure C.6: The intensity fitting results of neon at 7.8-11.1 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

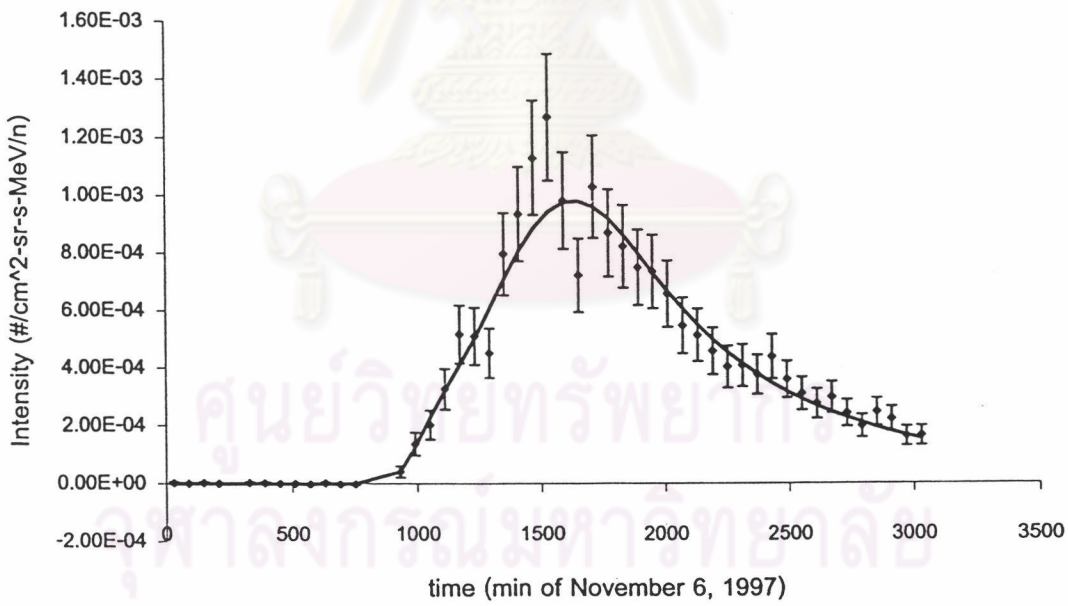


Figure C.7: The intensity fitting results of neon at 11.1-14.6 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

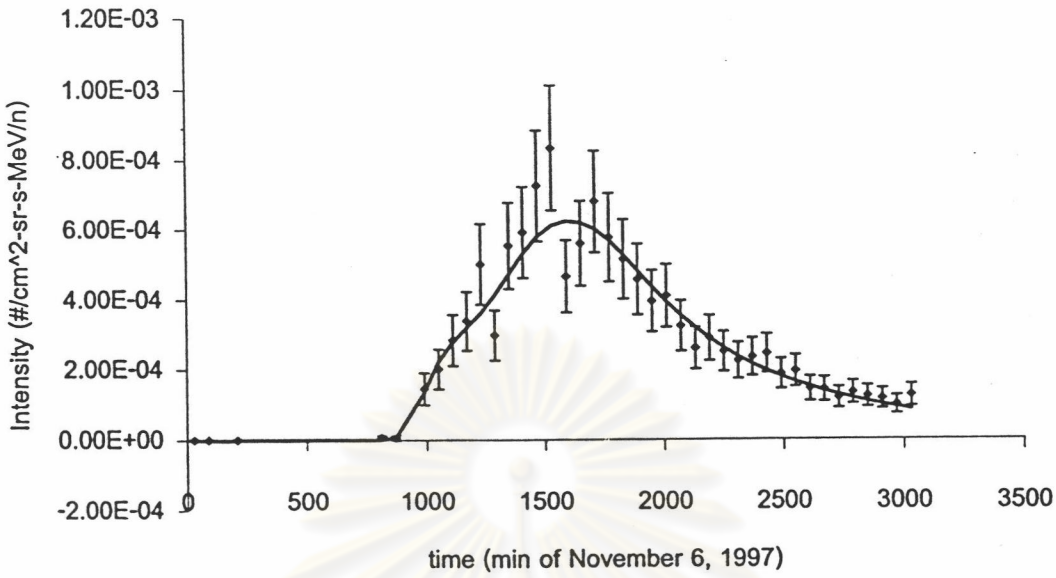


Figure C.8: The intensity fitting results of neon at 14.6-17.6 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

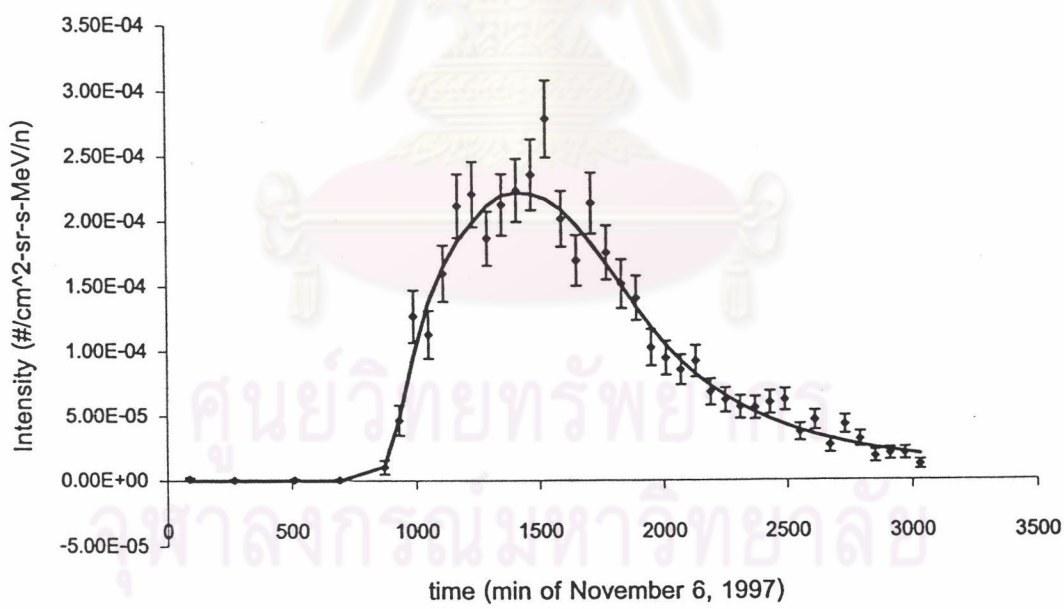


Figure C.9: The intensity fitting results of neon at 23.6-33.2 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

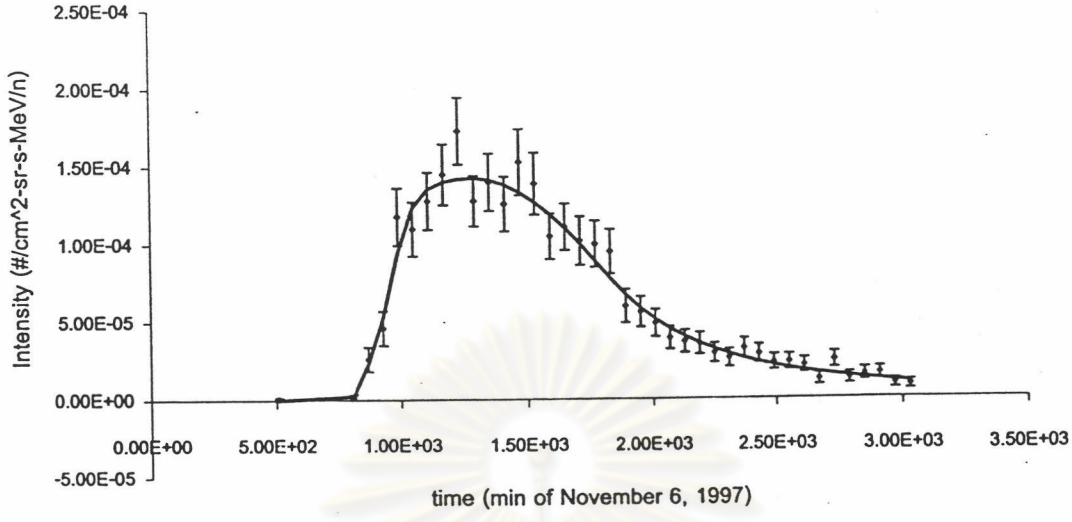


Figure C.10: The intensity fitting results of neon at 33.2-44.0 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

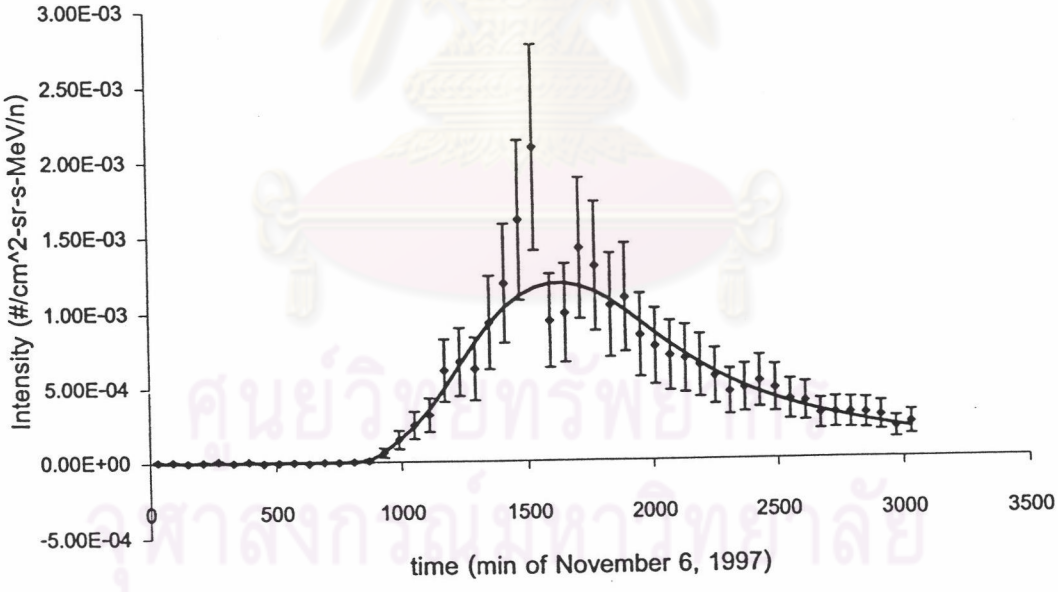


Figure C.11: The intensity fitting results of magnesium at 8.5-12.2 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

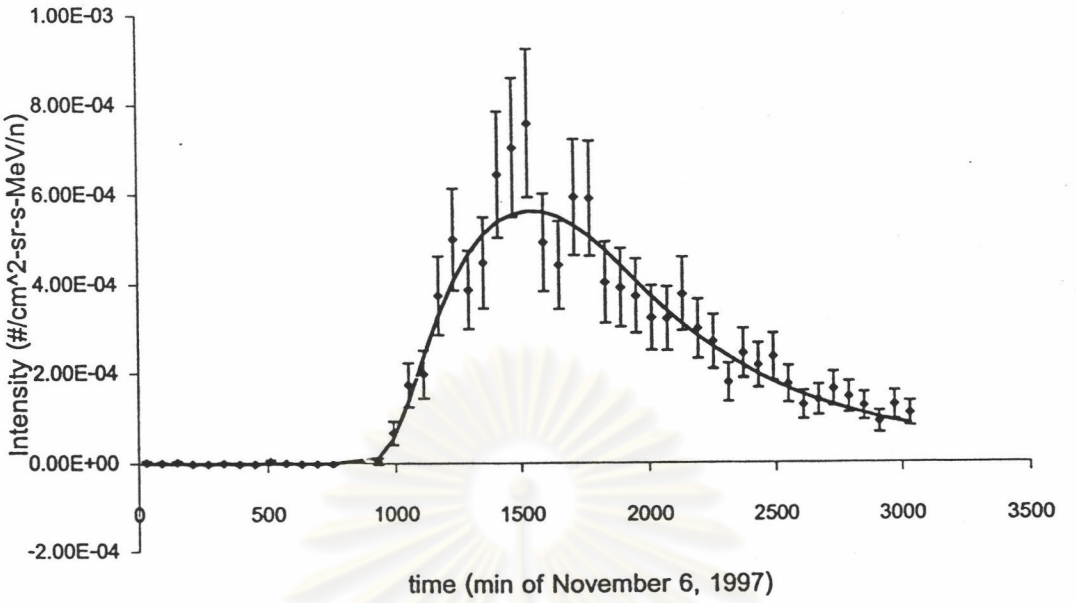


Figure C.12: The intensity fitting results of magnesium at 12.2-16.0 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

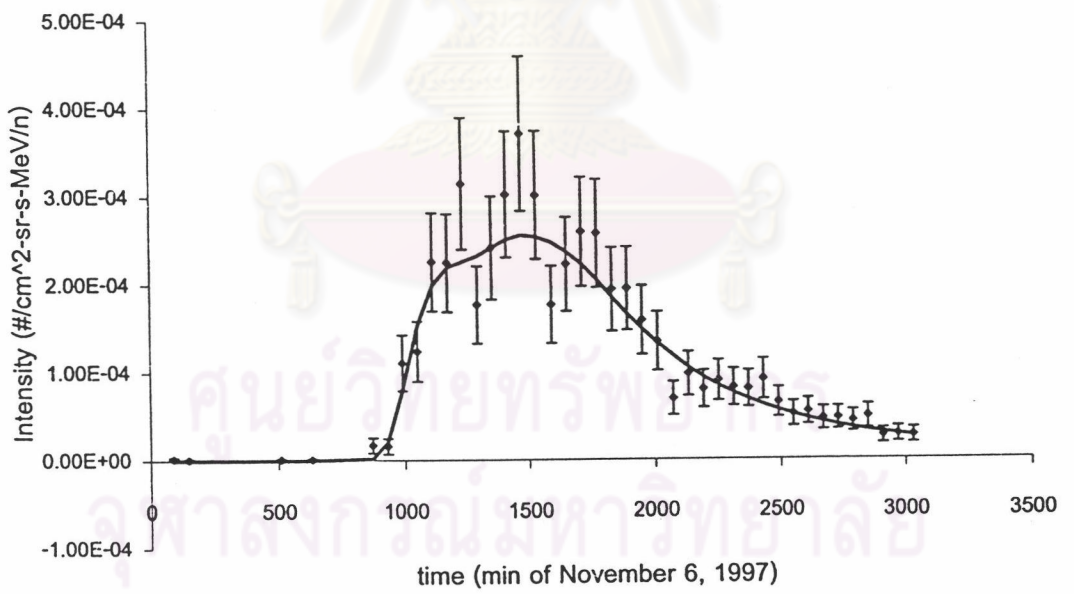


Figure C.13: The intensity fitting results of magnesium at 19.0-26.0 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

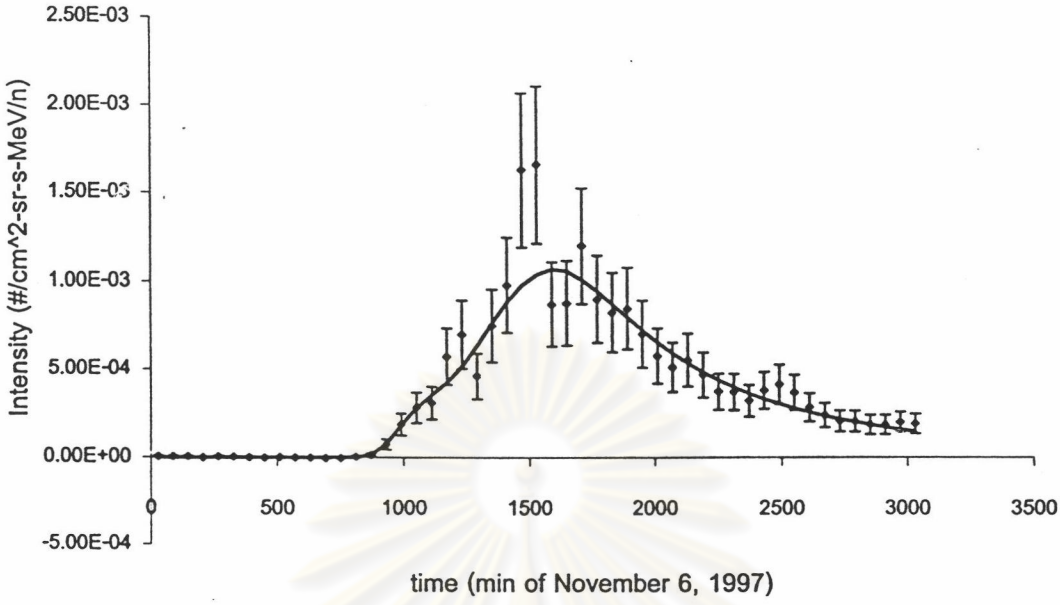


Figure C.14: The intensity fitting results of silicon at 9.0-13.0 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

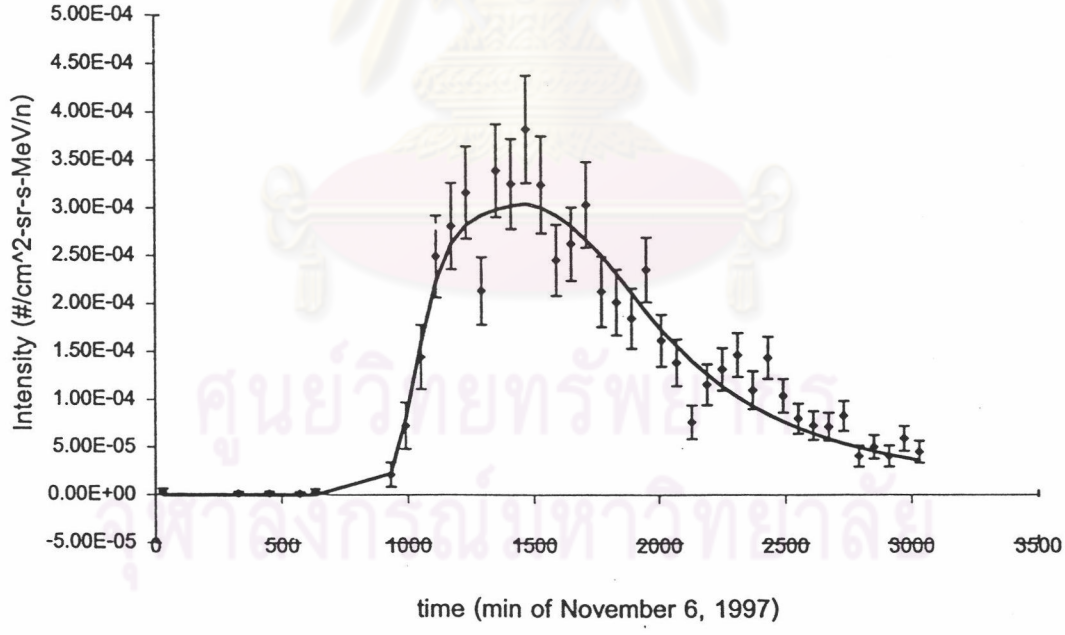


Figure C.15: The intensity fitting results of silicon at 17.3-20.8 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

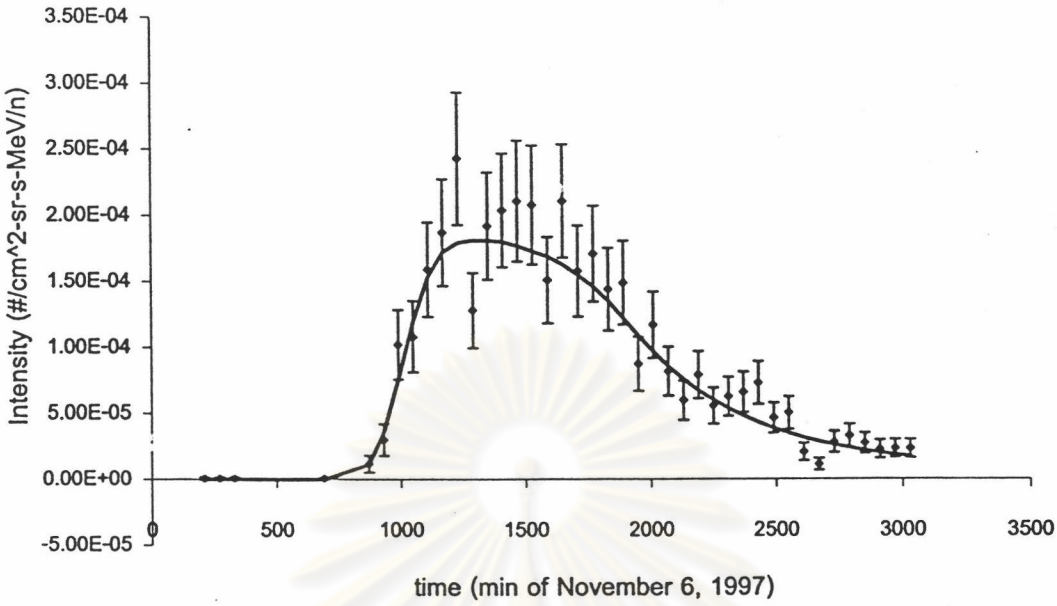


Figure C.16: The intensity fitting results of silicon at 20.8-28.1 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

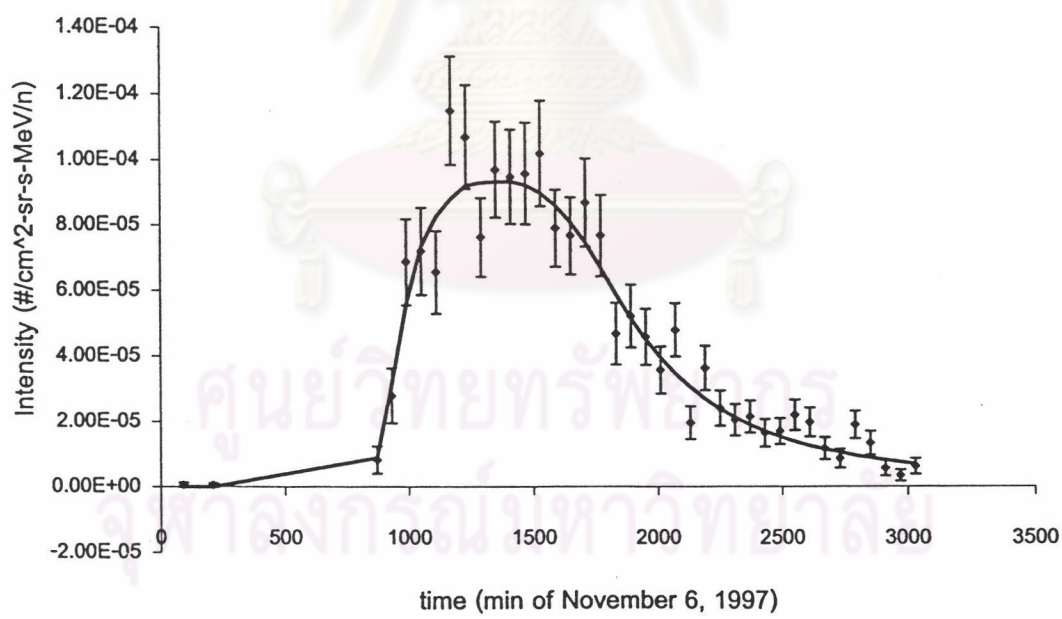


Figure C.17: The intensity fitting results of silicon at 28.1-39.8 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

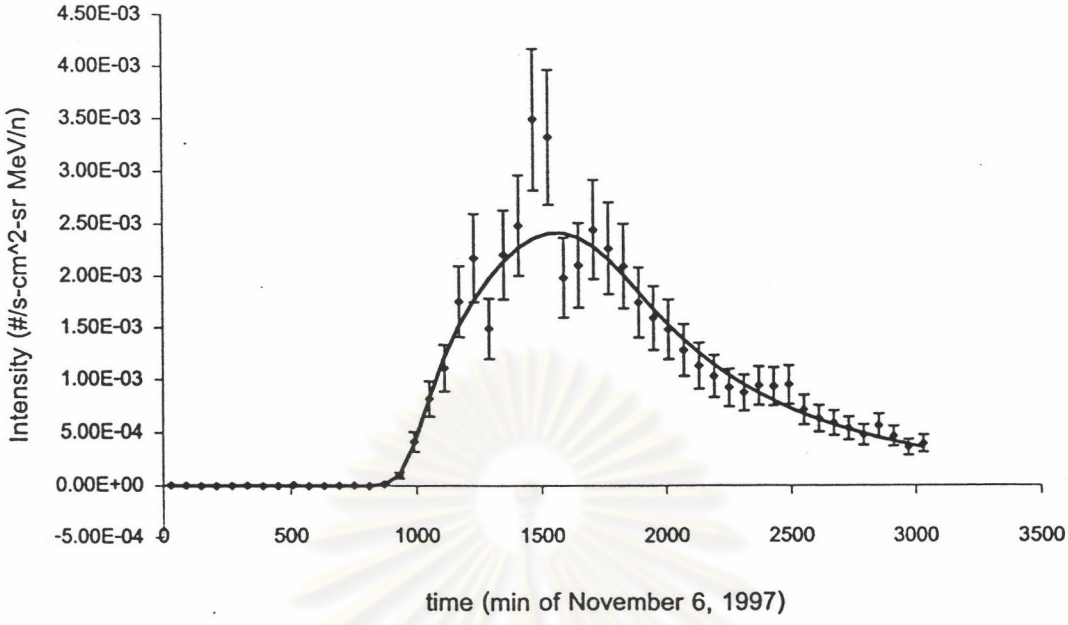


Figure C.18: The intensity fitting result of iron at 10.5-15.8 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

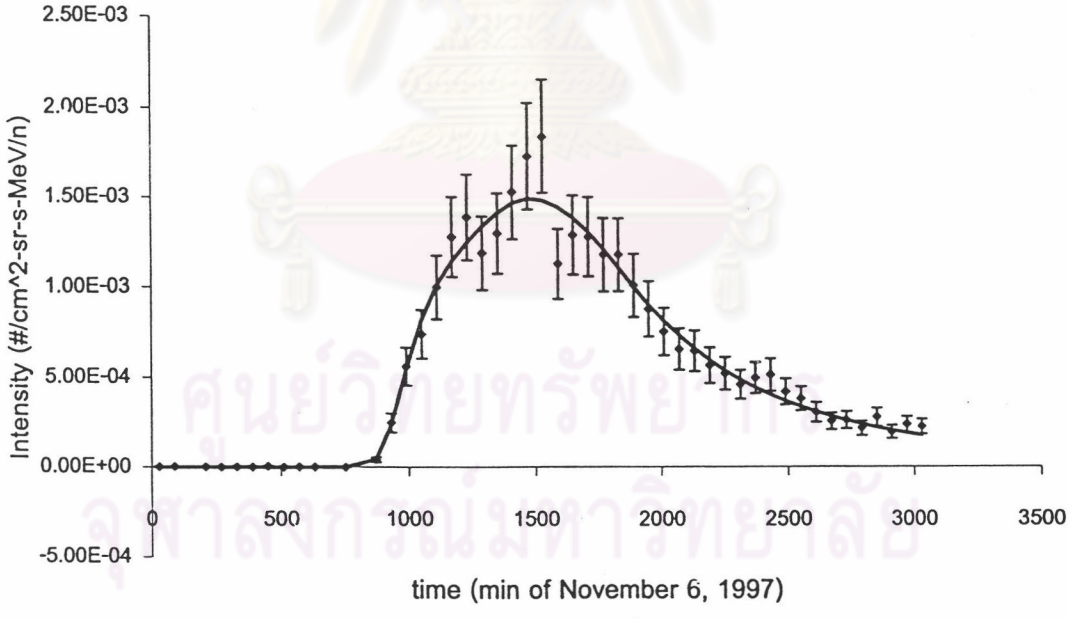


Figure C.19: The intensity fitting results of iron at 15.8-21.5 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

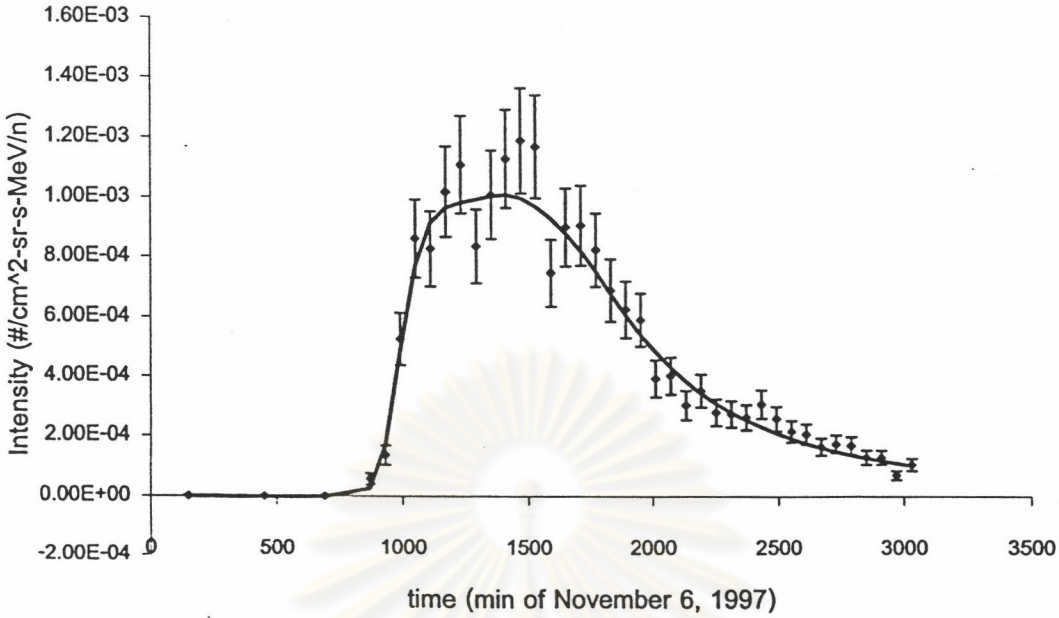


Figure C.20: The intensity fitting results of iron at 21.5-26.3 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

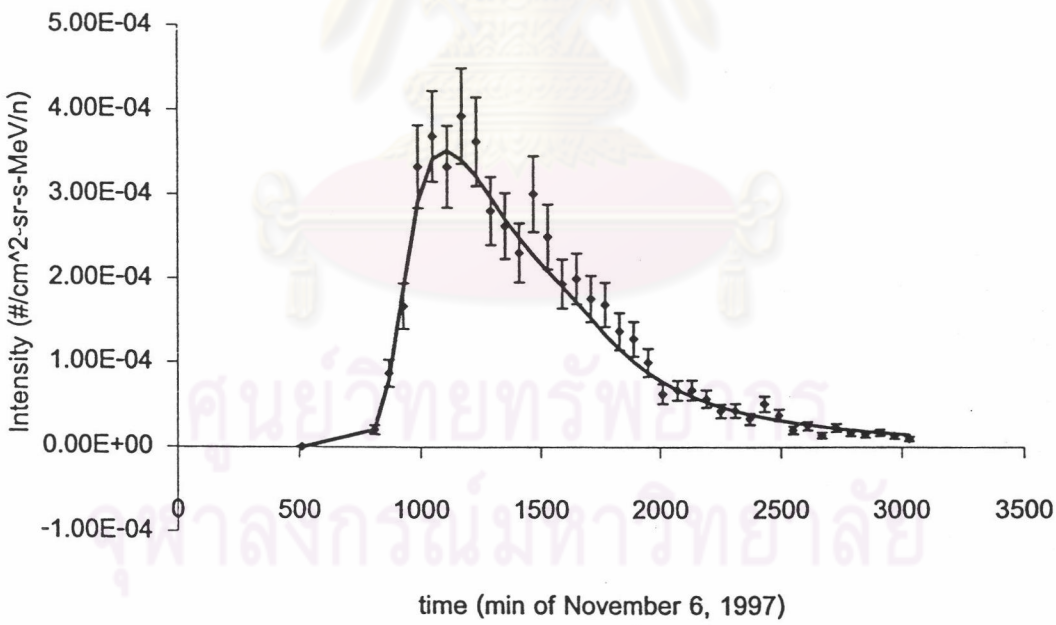
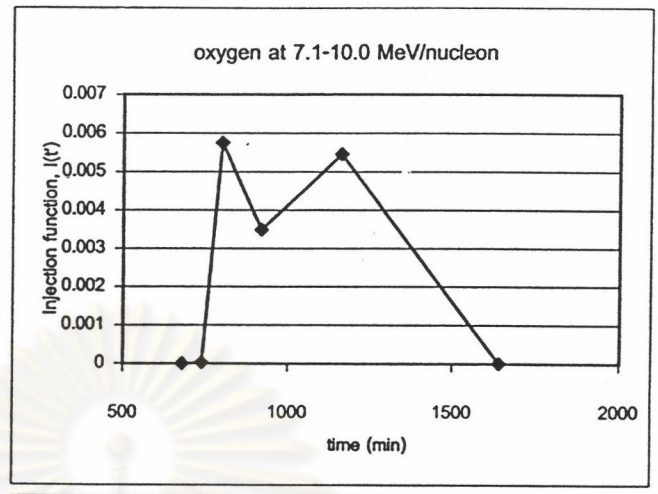


Figure C.21: The intensity fitting results of iron at 36.3-52.2 MeV/n from the SIS instrument on the ACE spacecraft on November 6, 1997.

Injection function of oxygen on November 6, 1997

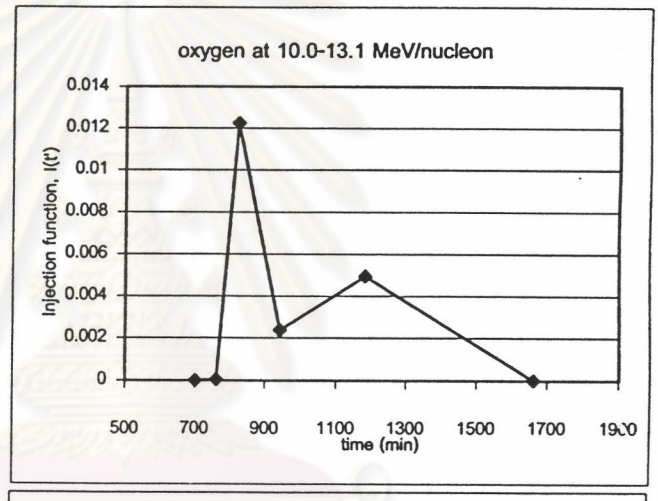
o97301s1o_0046

681	0	0
741	0.00002605	0.001189
801	0.0057555	0.002674
921	0.0034907	0.00215
1161	0.0054606	0.000708
1641	0	0



o97301s2o_0042

702	0	0
762	0.000041771	0.001732
822	0.012247	0.003655
942	0.002381	0.002529
1182	0.0049468	0.000706
1662	0	0



o97301s3o_0052

682	0	0
742	0.000060465	0.001943
802	0.0078872	0.003176
922	0.0032727	0.001896
1162	0.0055702	0.000555
1642	0	0

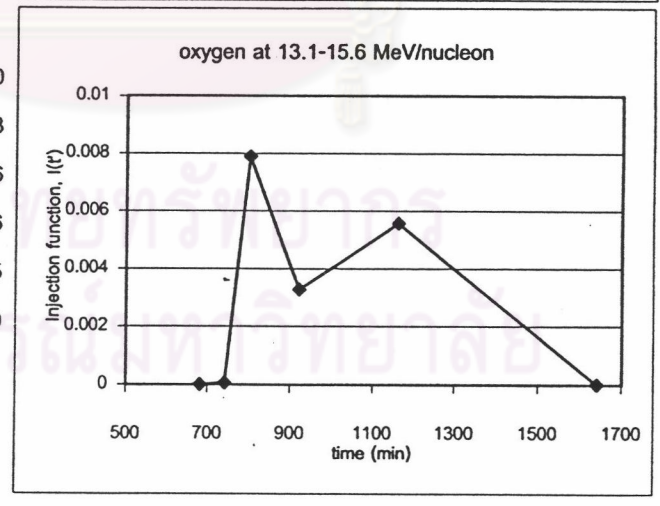
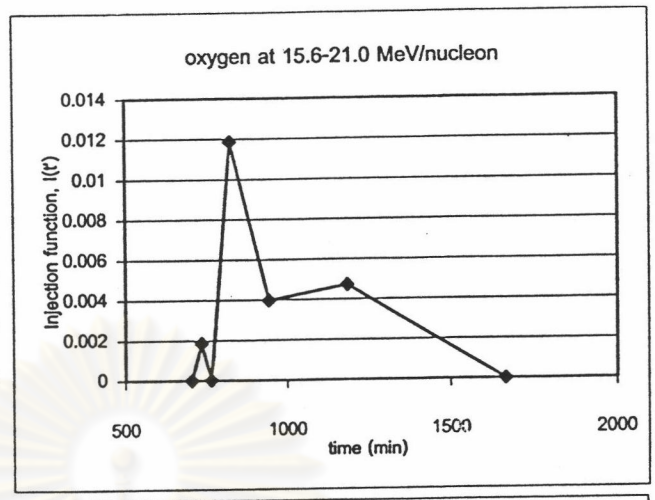


Figure C.22: The injection function of oxygen at 7.1-10.0, 10.0-13.1, and 13.1-15.6 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of oxygen on November 6, 1997

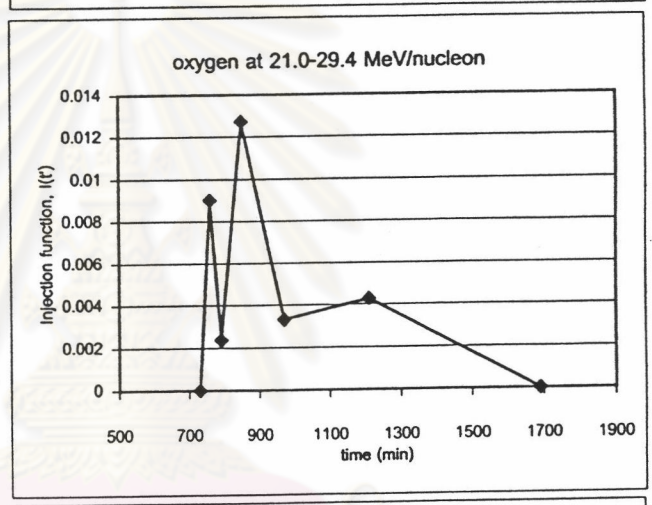
o97301s4o_0054

705	0	0
735	0.0018553	0.003519
765	0.000020276	0.006545
825	0.011857	0.004282
945	0.0039695	0.001882
1185	0.0047228	0.000465
1665	0	0



o97301s5o_0054

730	0	0
760	0.0090119	0.005483
790	0.0023723	0.010011
850	0.012736	0.006394
970	0.0033136	0.002464
1210	0.0042788	0.000545
1690	0	0



o97301s6o_0054

725	0	0
755	0.01897	0.007906
785	0.00094381	0.013201
845	0.018786	0.008369
965	0.0014937	0.003079
1205	0.0041339	0.000626
1685	0	0

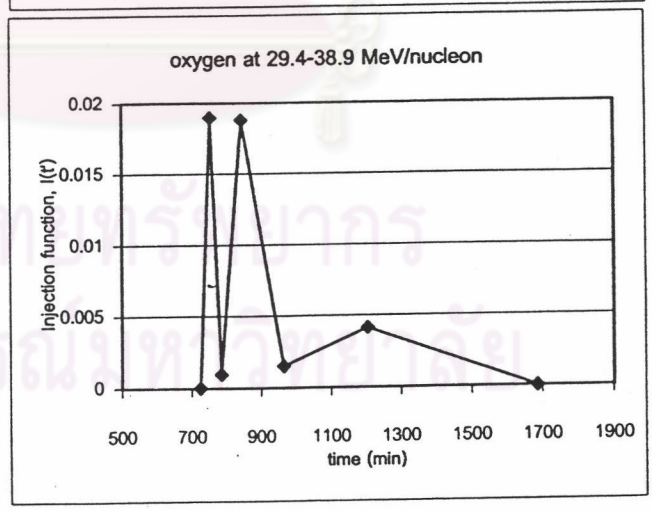
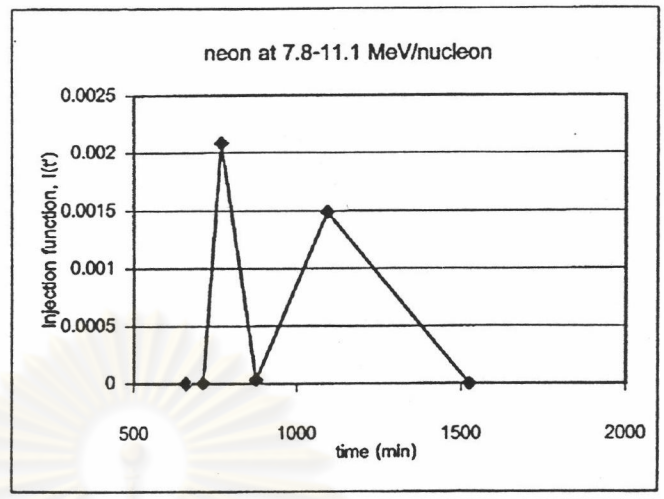


Figure C.23: The injection function of oxygen at 15.6-21.0, 21.0-29.4, and 29.4-38.9 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of neon on November 6, 1997

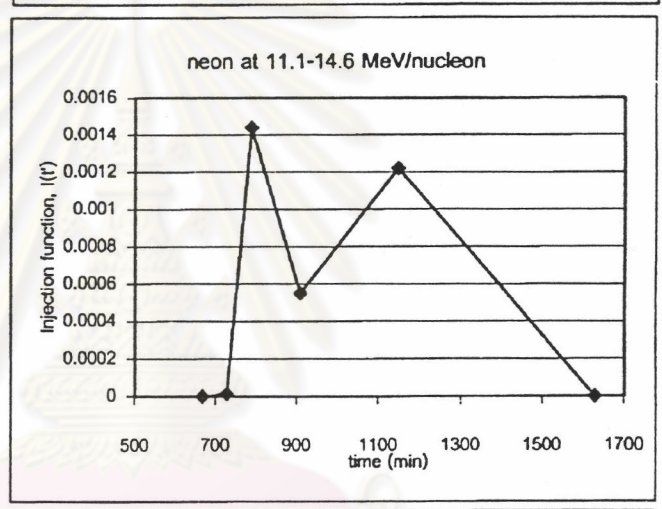
ne97301s1o_00474

661	0	0
715	3.3183E-06	0.00034615
769	0.0020841	0.00069766
877	0.000034052	0.0004949
1093	0.0014898	0.0001568
1525	0	0



ne97301s2o_00495

669	0	0
729	0.000014111	0.00053231
789	0.0014382	0.00078153
909	0.00054887	0.00039503
1149	0.0012183	0.0001053
1629	0	0



ne97301s3o_00541

695	0	0
755	0.00021335	0.00041328
815	0.001974	0.00068553
935	0.00023122	0.00043156
1175	0.0011658	0.00012229
1655	0	0

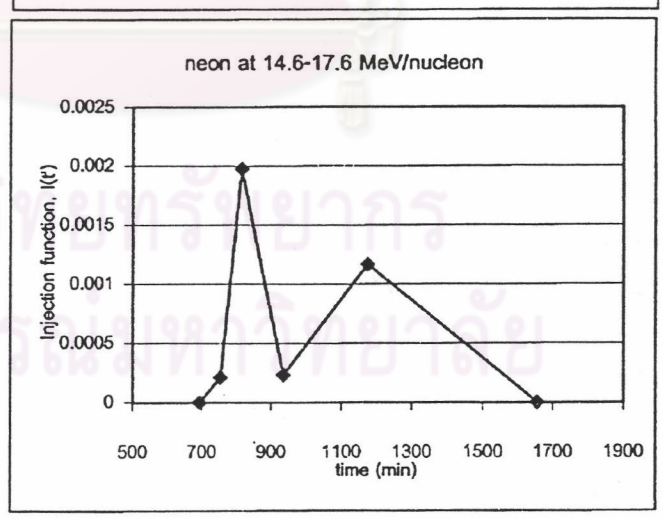
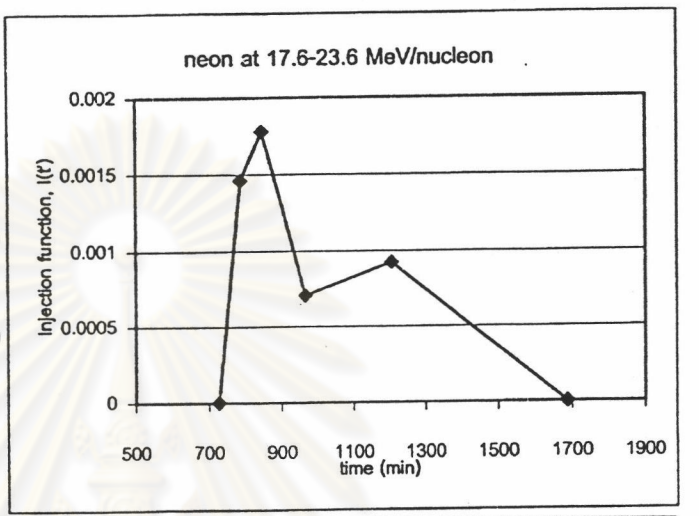


Figure C.24: The injection function of neon at 7.8-11.1, 11.1-14.6, and 14.6-17.6 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of neon on November 6, 1997

ne97301s4o_00537

727	0	0
787	0.0014608	0.00081217
847	0.00178	0.0011113
967	0.00070867	0.00051563
1207	0.00092498	0.00012491
1687	0	0



ne97301s5o_00413

657	0	0
717	0.000024173	0.00056166
777	0.0032278	0.00084687
897	0.00095346	0.00040189
1137	0.0010378	0.000091666
1617	0	0

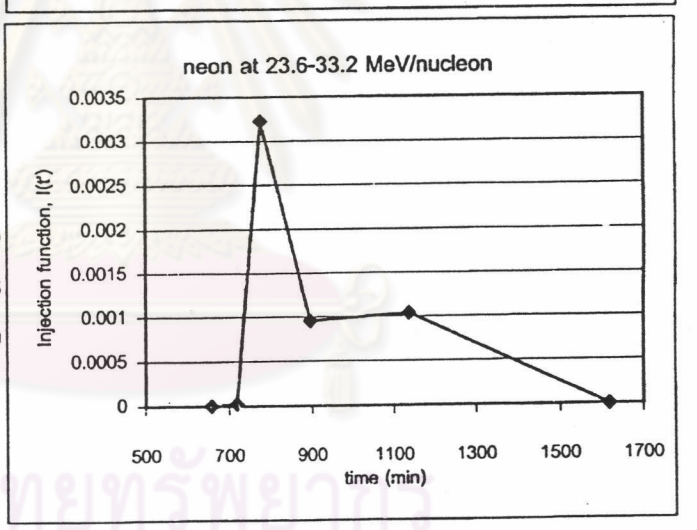
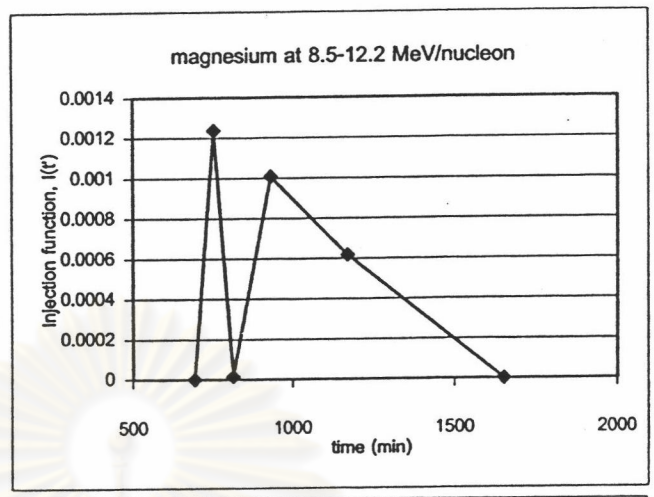


Figure C.25: The injection function of neon at 17.6-23.6 and 23.6-33.2 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of magnesium on November 6, 1997

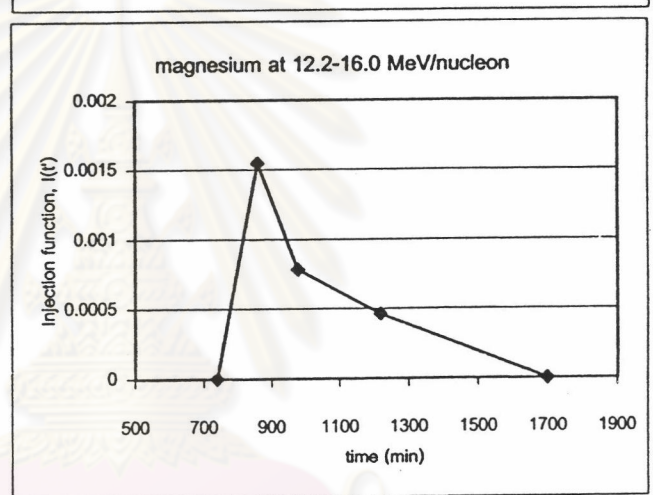
mg97301s1o_0043

692	0	0
752	0.0012366	0.00061365
812	0.000013864	0.00096233
932	0.001008	0.00057374
1172	0.00061847	0.0001518
1652	0	0



mg97301s2o_0044

739	0	0
859	0.0015453	0.00030223
979	0.00078182	0.00036526
1219	0.00046062	0.00010679
1699	0	0



mg97301s3o_0041

739	0	0
859	0.0016502	0.00027885
979	0.00083283	0.00028304
1219	0.00043463	0.000074343
1699	0	0

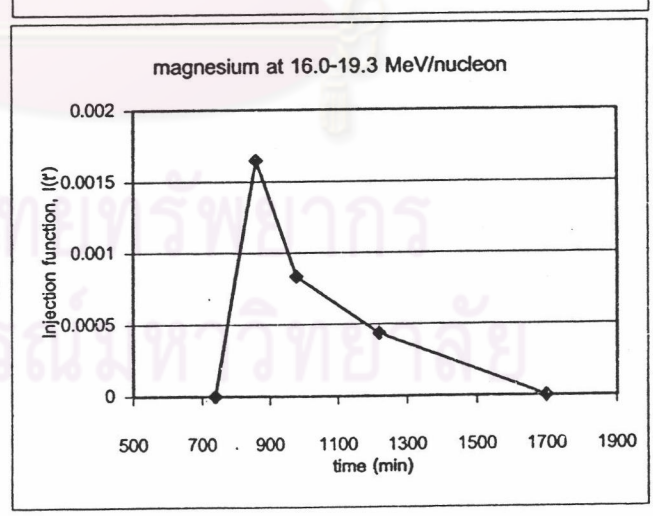


Figure C.26: The injection function of magnesium at 8.5-12.2, 12.2-16.0, and 16.0-19.3 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of magnesium on November 6, 1997

mg97301s4o_0044

737	0	0
791	0.00079191	0.00073941
845	0.003069	0.0012264
953	7.8723E-06	0.00062623
1169	0.00073262	0.00013952
1601	0	0

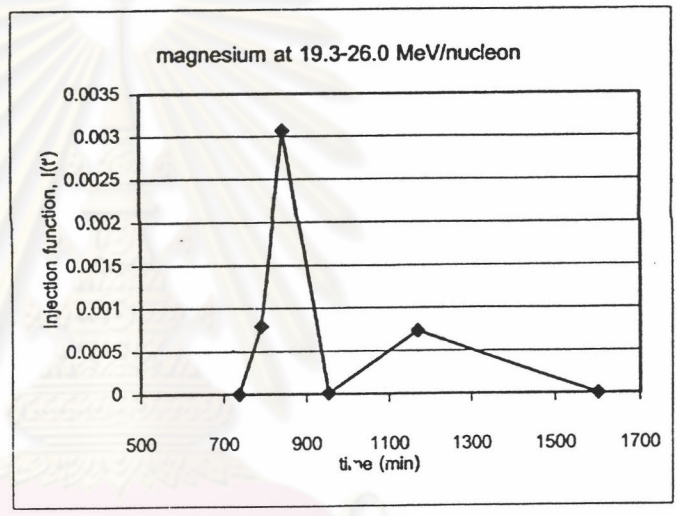
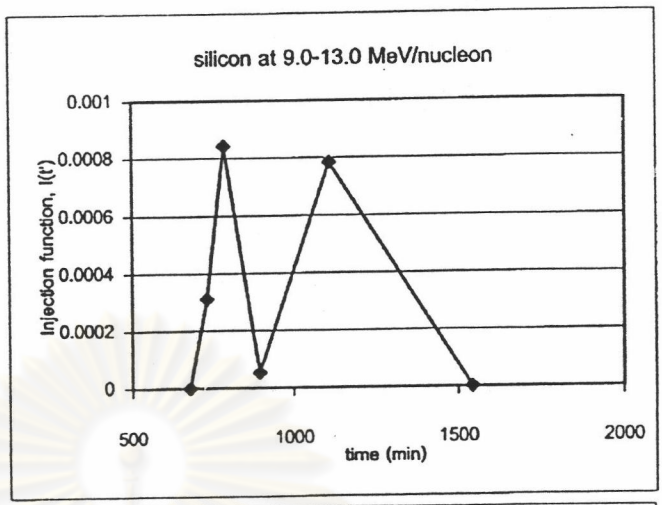


Figure C.27: The injection function of magnesium at 19.3-26.0 MeV/n of the solar event on November 6, 1997.

Injection function of silicon on November 6, 1997

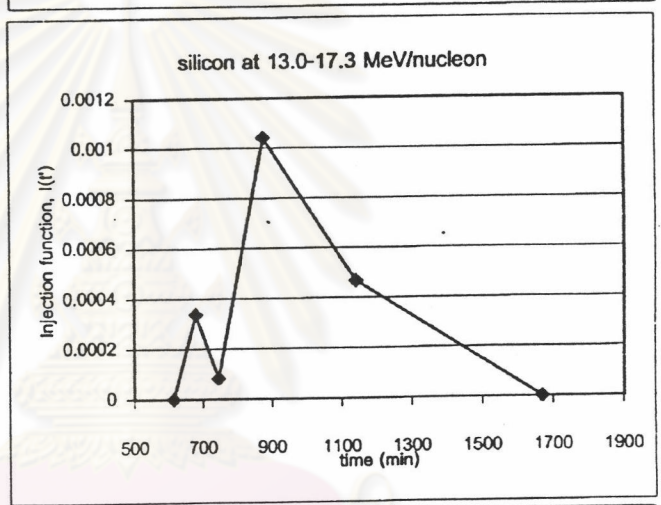
si97301s1o_0051

679	0	0
733	0.00031118	0.0002922
787	0.00084263	0.00050679
895	0.000052596	0.00031718
1111	0.00078348	0.000093774
1543	0	0



si97301s2c_0042

614	0	0
680	0.00033526	0.00031294
746	0.00008153	0.00044415
878	0.0010412	0.00027469
1142	0.00046684	0.000079006
1670	0	0



si97301s3o_0048

739	0	0
859	0.001915	0.00026568
979	0.00033945	0.0002501
1219	0.00043812	0.000064366
1699	0	0

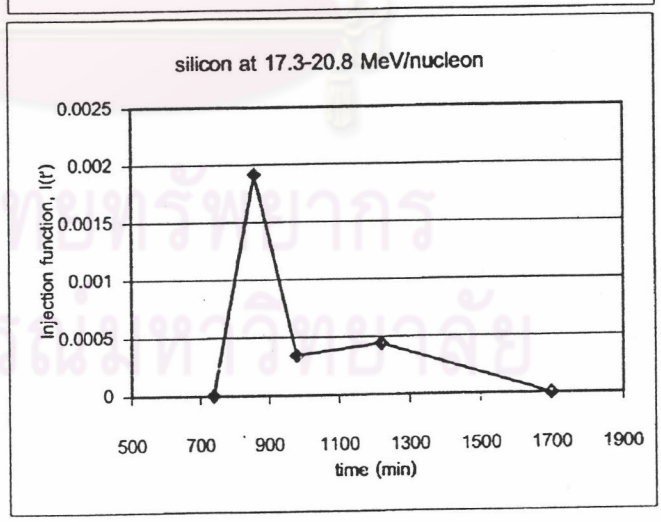
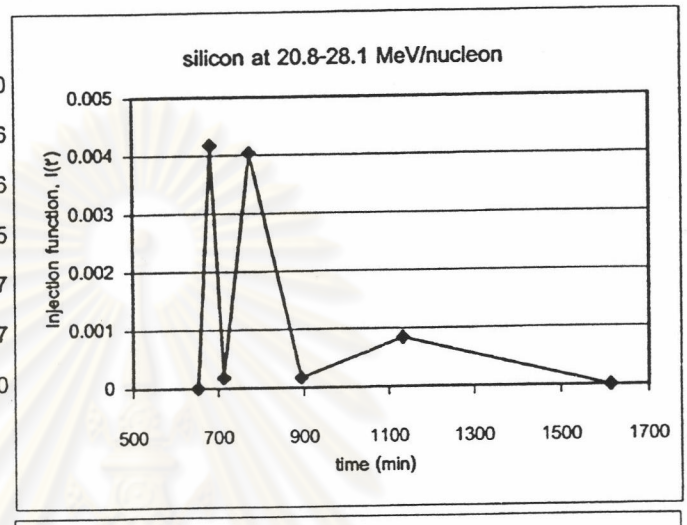


Figure C.28: The injection function of silicon at 9.0-13.0, 13.0-17.3, and 17.3-20.8 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of silicon on November 6, 1997

si97301s4o_0023

653	0	0
683	0.0041734	0.0075046
713	0.00017543	0.011046
773	0.0040376	0.0047355
893	0.00016605	0.0011627
1133	0.00084131	0.00017707
1613	0	0



si97301s5o_0024

684	0	0
738	0.0066691	0.0019155
792	0.000076608	0.0021787
900	0.00076736	0.00075755
1116	0.00085039	0.00013833
1548	0	0

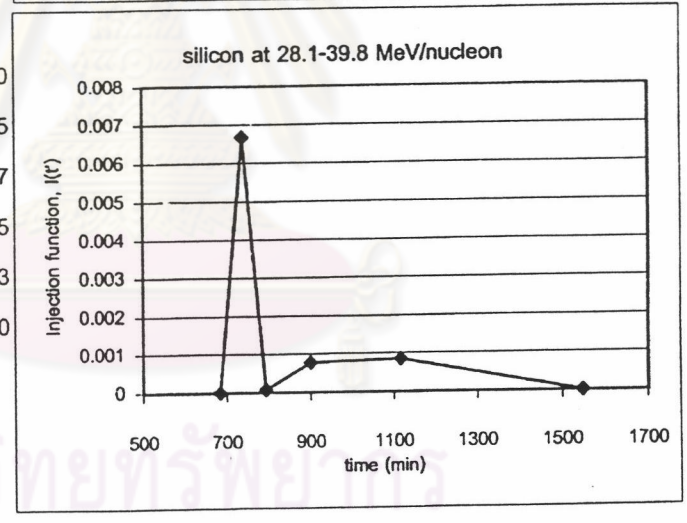
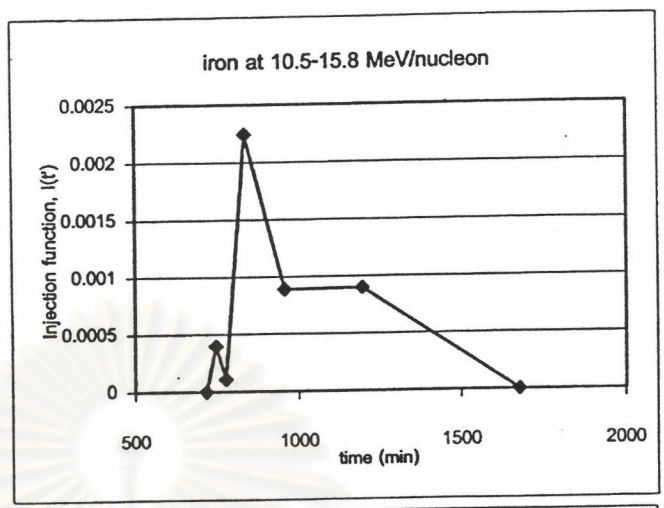


Figure C.29: The injection function of silicon at 20.8-28.1 and 28.1-39.8 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of iron on November 6, 1997

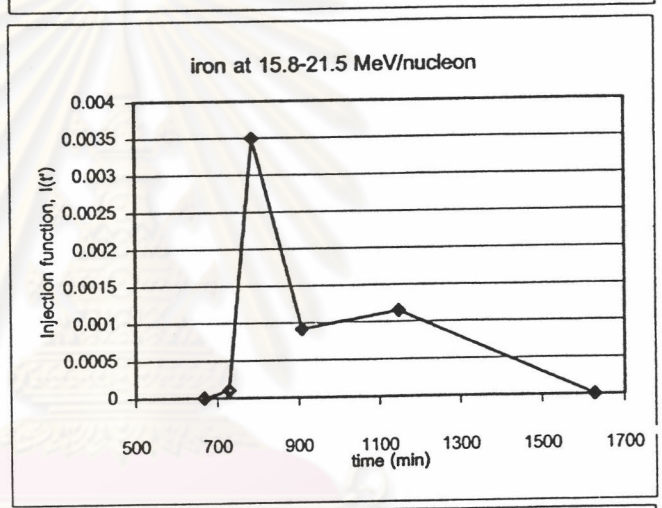
fe97301s1o_00058

717	0	0
747	0.000395	0.000521
777	0.00010684	0.001003
837	0.0022427	0.00083
957	0.00089188	0.000461
1197	0.00090032	0.000122
1677	0	0



fe97301s2o_005

668	0	0
728	0.000101	0.000328
788	0.0034932	0.000758
908	0.0009154	0.000509
1148	0.0011493	0.000136
1628	0	0



fe97301s3o_0064

734	0	0
794	0.001188	0.000533
854	0.0042789	0.000962
974	0.00077148	0.00051
1214	0.00099768	0.000119
1694	0	0

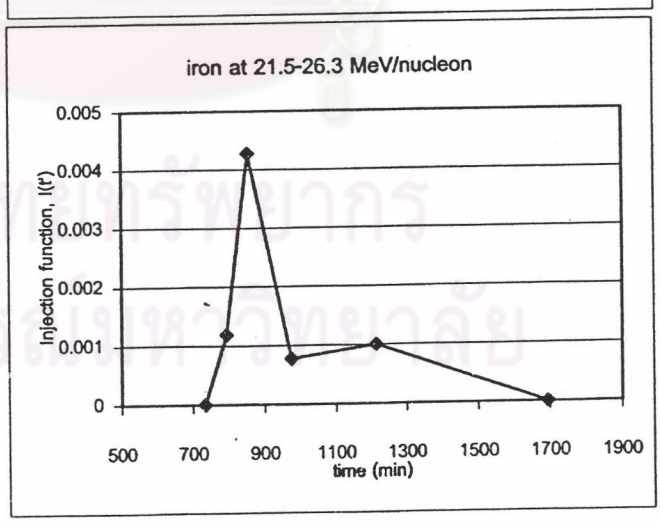
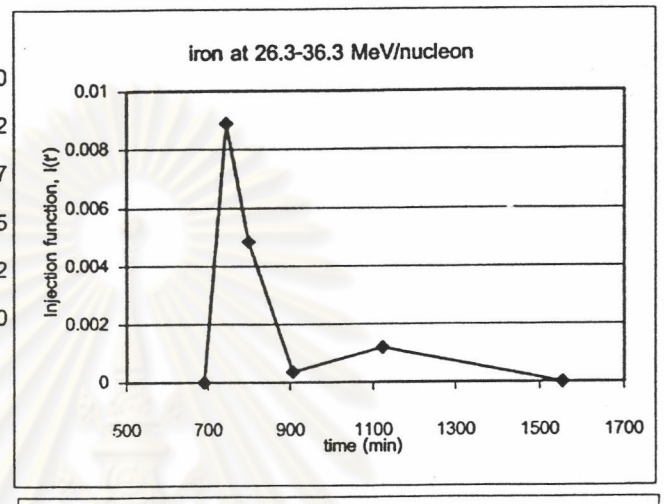


Figure C.30: The injection function of iron at 10.5-15.8, 15.8-21.5, and 21.5-26.3 MeV/n, respectively, of the solar event on November 6, 1997.

Injection function of iron on November 6, 1997

fe97301s4o_0033

691	0	0
745	0.0088825	0.0012242
799	0.0048457	0.0014717
907	0.00035257	0.00054515
1123	0.0011884	0.000098942
1555	0	0



fe97301s5o_0038

660	0	0
690	0.0032225	0.0025939
720	0.0005994	0.0043909
780	0.0067655	0.0024649
900	0.00082776	0.00077861
1140	0.00056246	0.00013502
1620	0	0

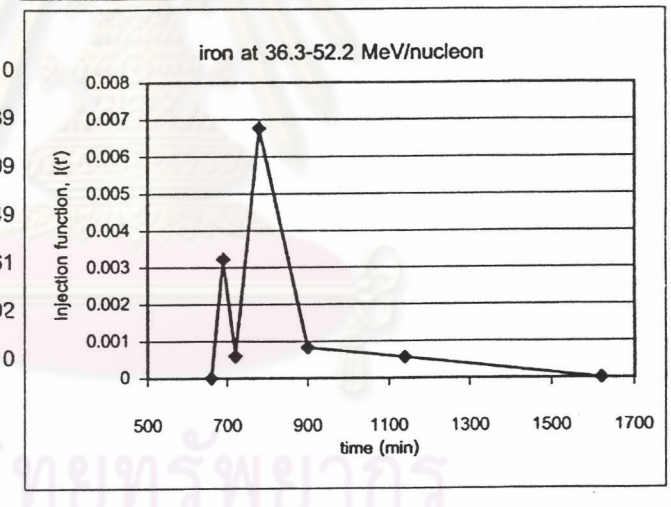


Figure C.31: The injection function of iron at 26.3-36.3 and 36.3-52.2 MeV/n, respectively, of the solar event on November 6, 1997.

Appendix D
Data That Could Not Be Fitted Well

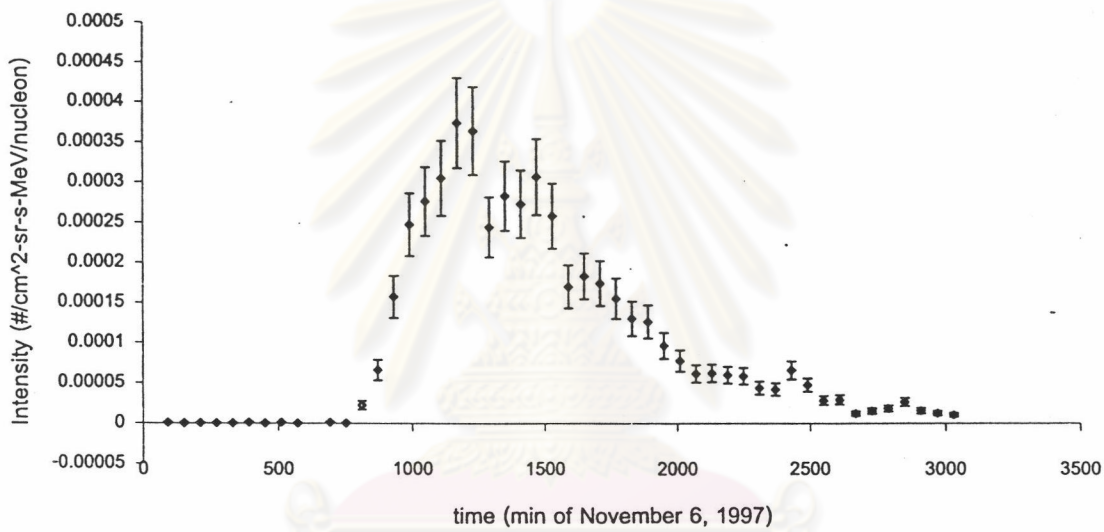


Figure D.1: The intensity data with their uncertainties of oxygen at 38.9-63.8 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

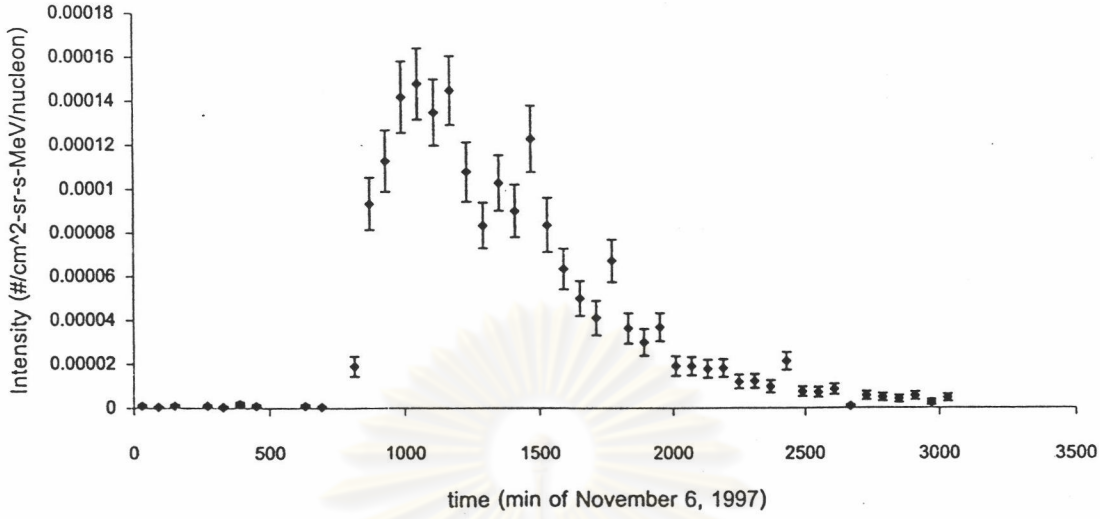


Figure D.2: The intensity data with their uncertainties of oxygen at 63.8-89.8 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

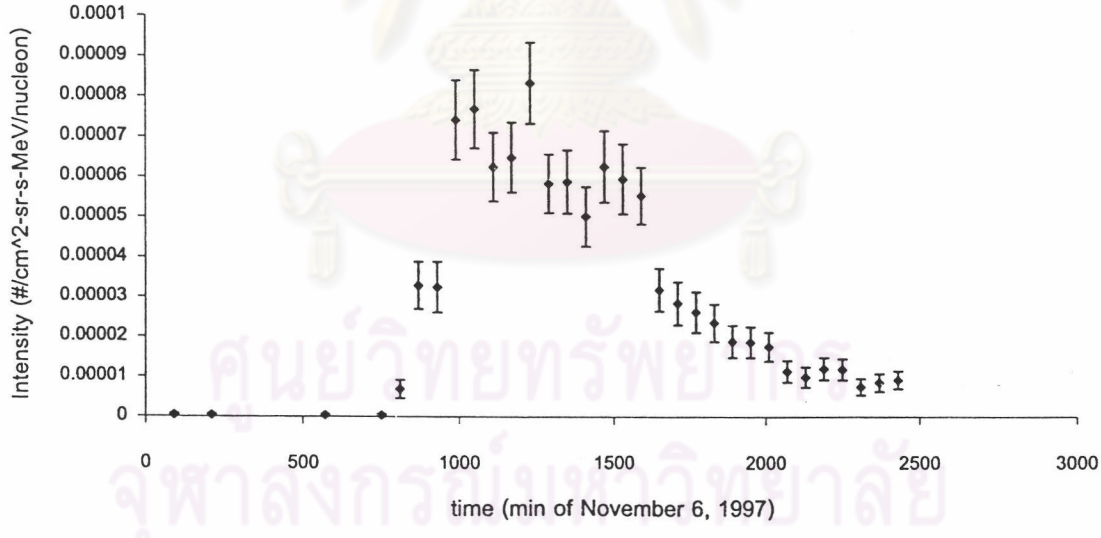


Figure D.3: The intensity data with their uncertainties of neon at 44.0-72.2 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

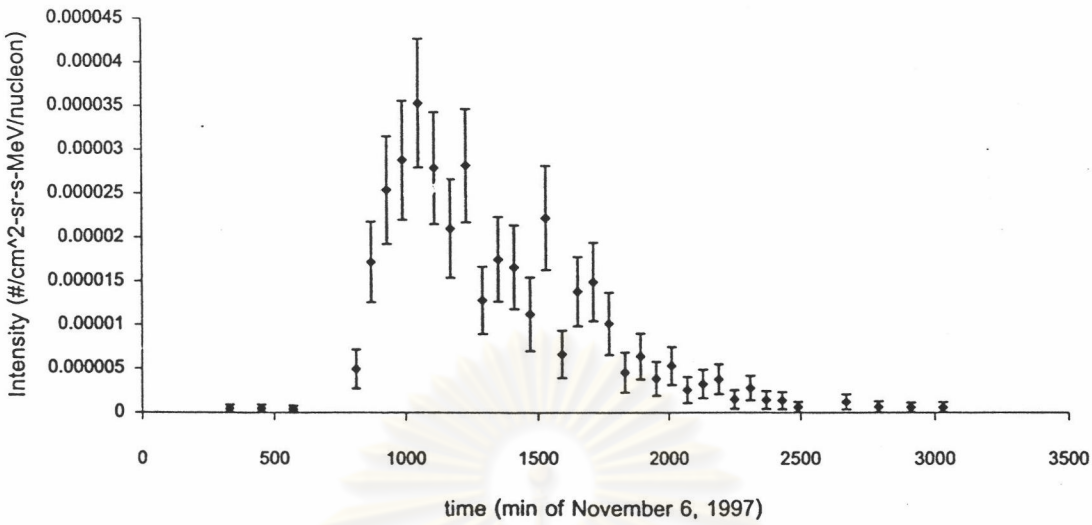


Figure D.4: The intensity data with their uncertainties of neon at 72.2-101.8 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because the fluctuations in the data were too great.

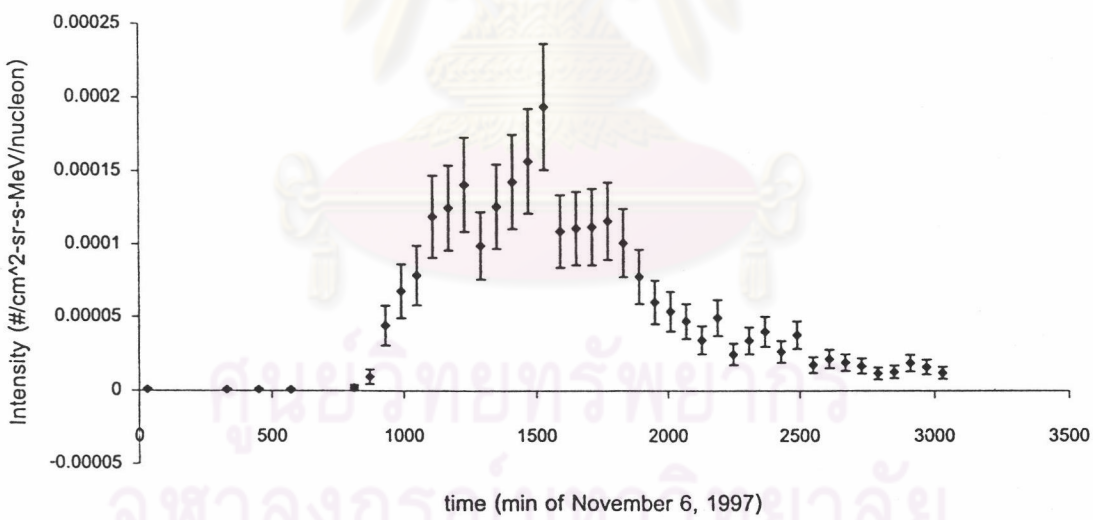


Figure D.5: The intensity data with their uncertainties of magnesium at 26.0-36.6 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

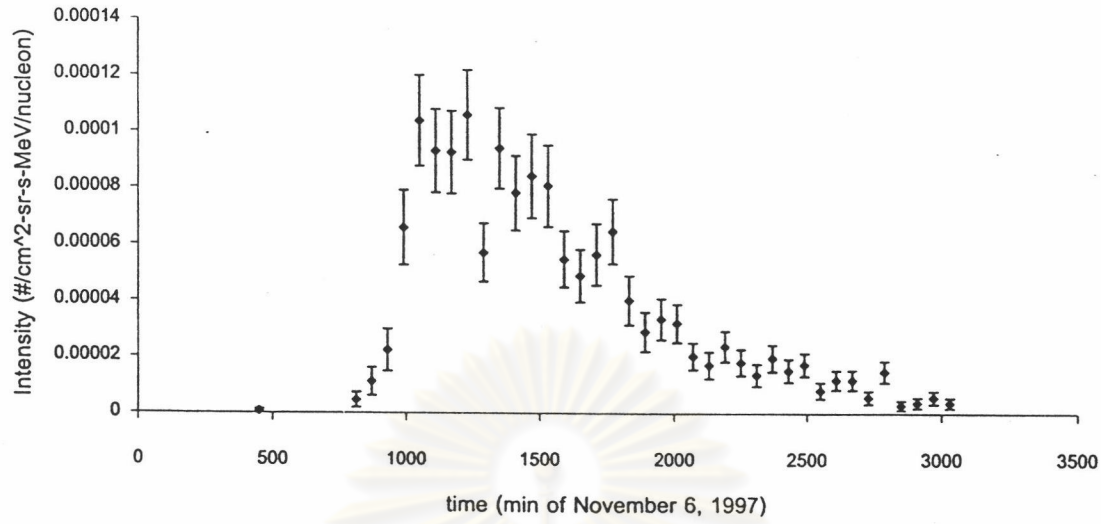


Figure D.6: The intensity data with their uncertainties of magnesium at 36.6-48.6 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

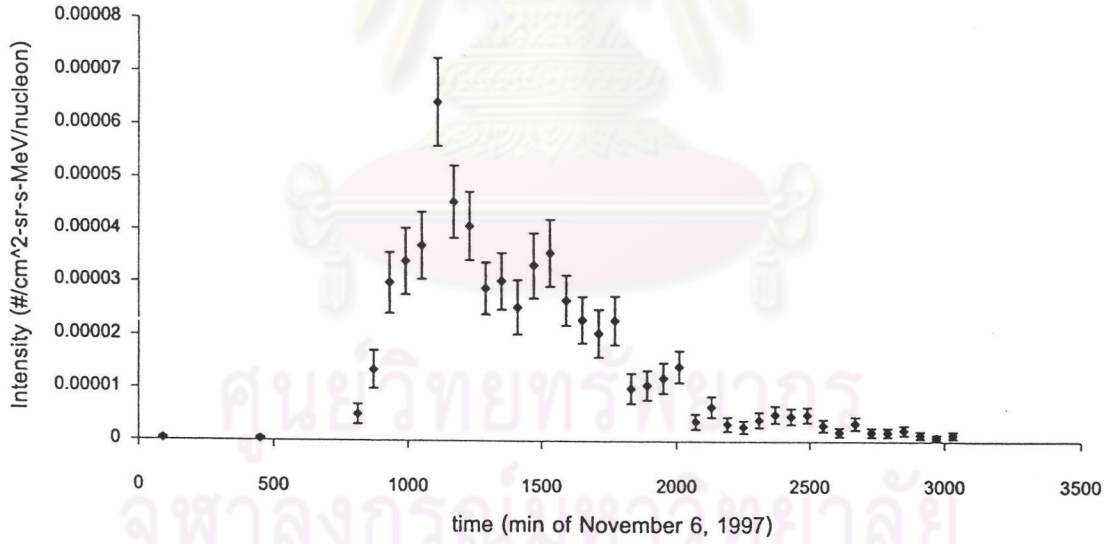


Figure D.7: The intensity data with their uncertainties of magnesium at 48.6-80.0 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

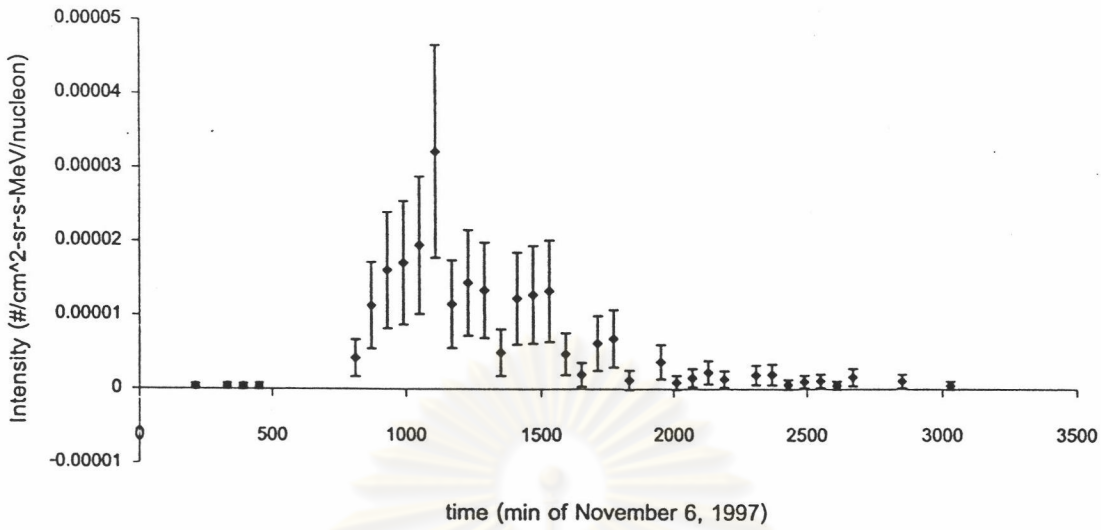


Figure D.8: The intensity data with their uncertainties of magnesium at 80.0-112.9 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because the fluctuations in the data were too great.

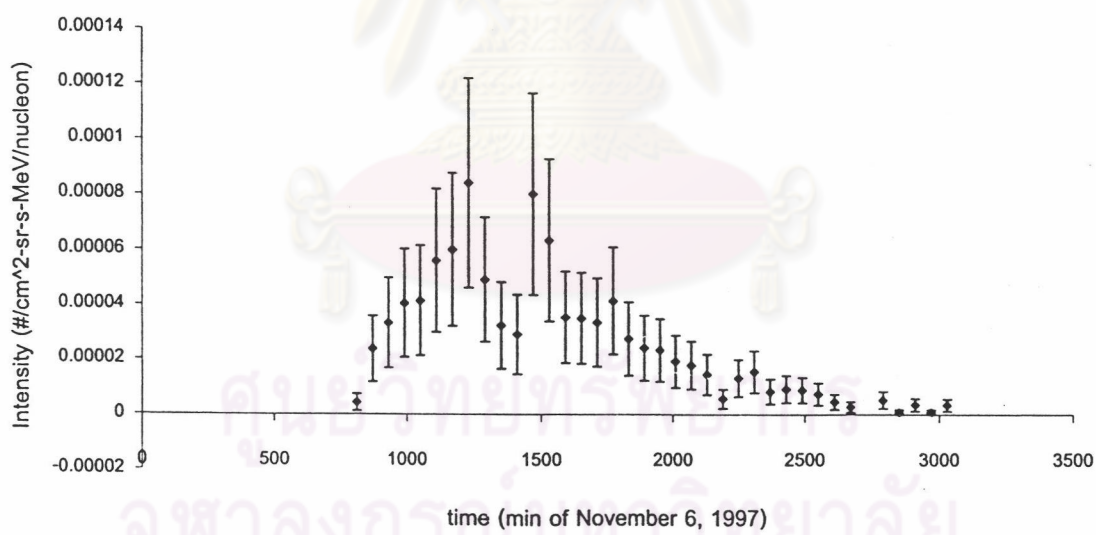


Figure D.9: The intensity data with their uncertainties of silicon at 39.8-52.9 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because the fluctuations in the data were too great.

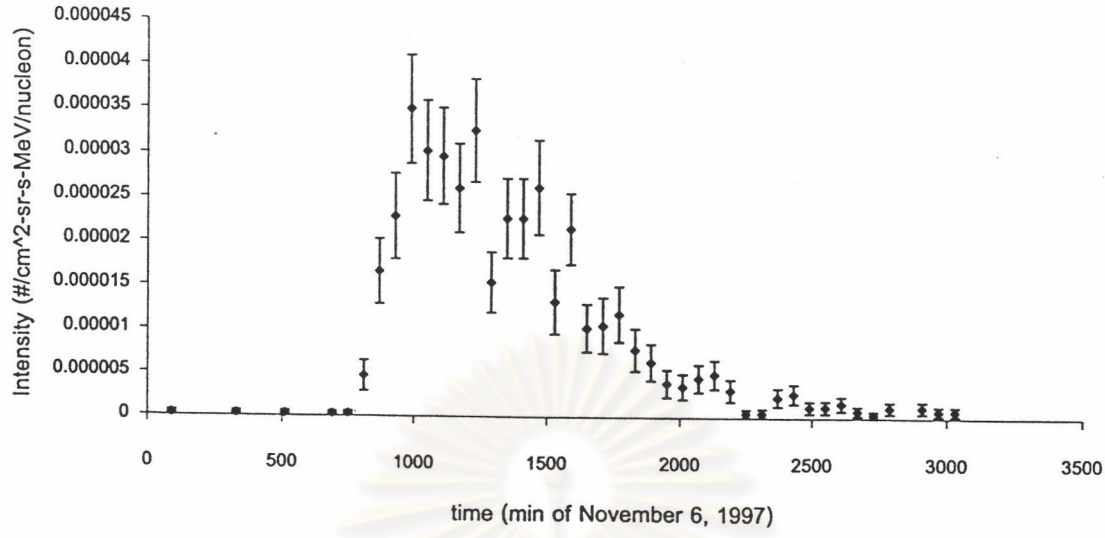


Figure D.10: The intensity data with their uncertainties of silicon at 52.9-87.1 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

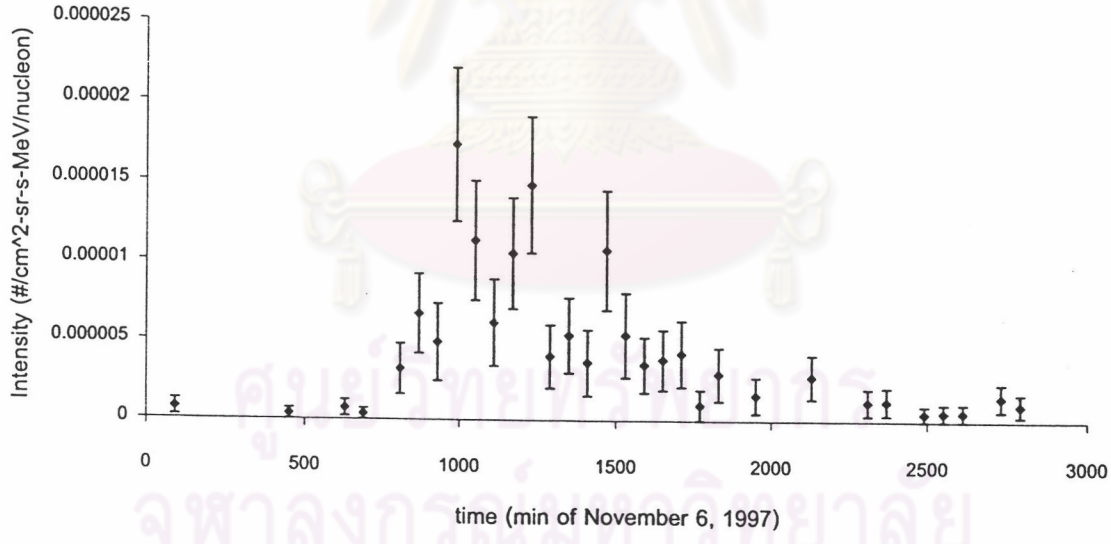


Figure D.11: The intensity data with their uncertainties of silicon at 87.1-123.2 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because the fluctuations in the data were too great.

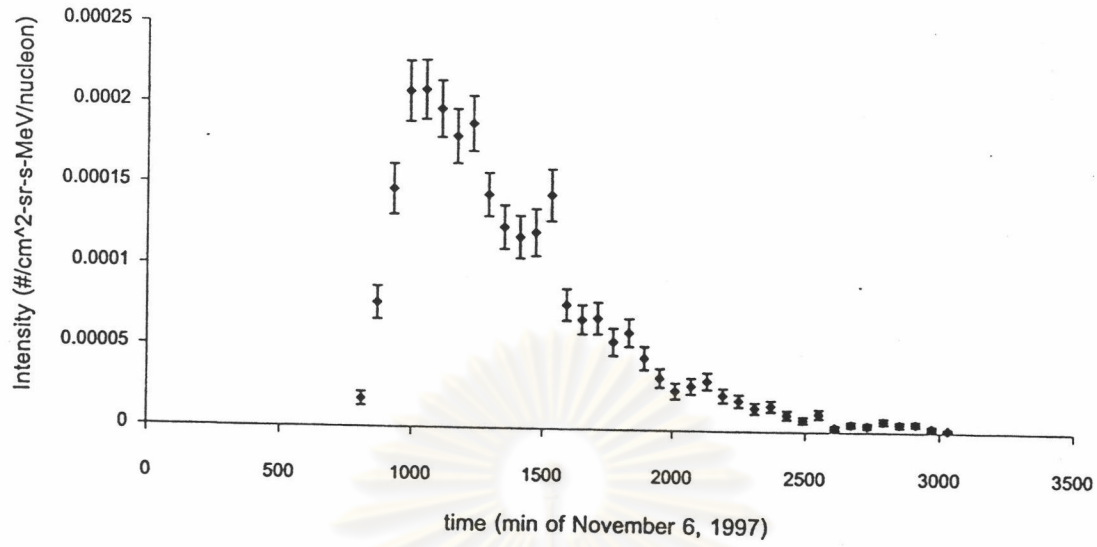


Figure D.12: The intensity data with their uncertainties of iron at 52.2-70.2 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because there was more than one minimum in χ^2 vs. λ.

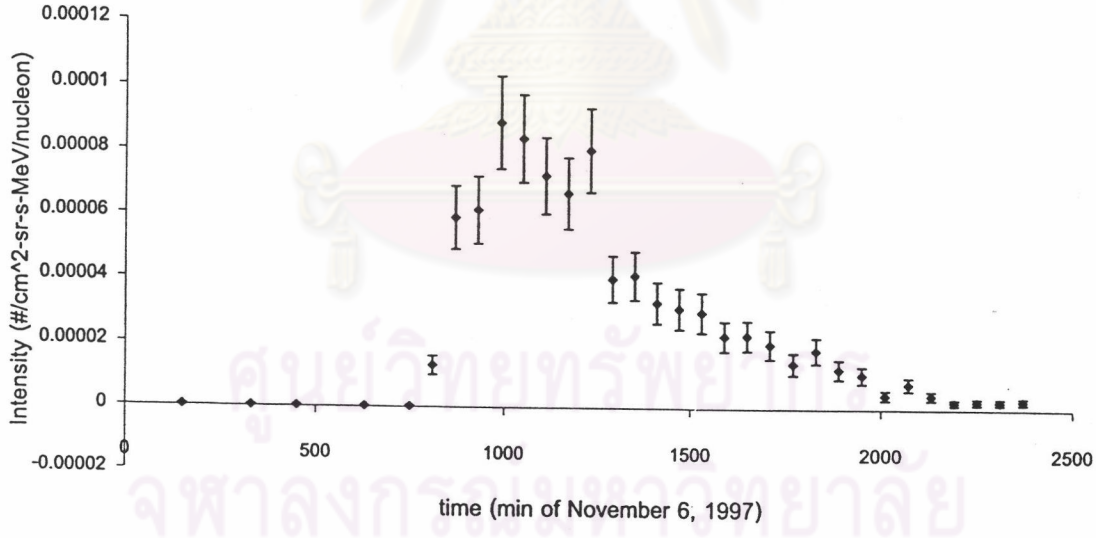


Figure D.13: The intensity data with their uncertainties of iron at 70.2-117.5 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because the fluctuations in the data were too great.

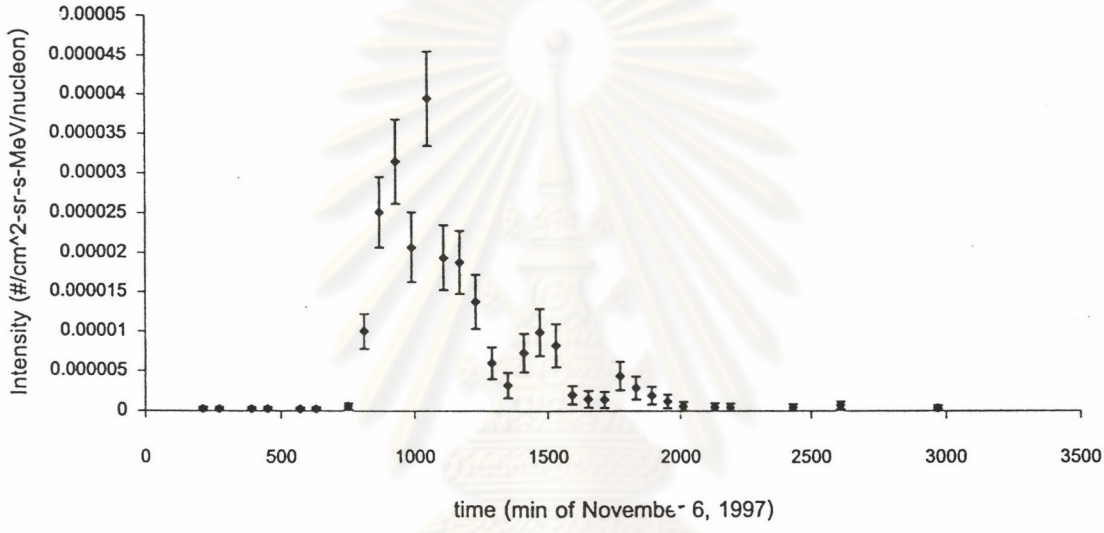


Figure D.14: The intensity data with their uncertainties of iron at 117.5-167.7 MeV/nucleon from the SIS instrument on the ACE spacecraft on November 6, 1997. These data could not be fitted because the fluctuations in the data were too great.

Appendix E
Wind Program for Simulation

```
/* wind.c (stvd) -- January 22, 2001
     Added another argument to mudot() (called in elements()).
   wind.c (stvd) -- September 12, 2000
     Modified to remove initial call to printout() (to avoid repetition
     with last output of previous stage) and to increase the maximum
     ntotal to 10000.
   wind.c (stvd) -- March 6, 2000
     Now call a new routine, enratio() -> energy ratio, in elements().
     Added #define EXPAND (expanding inner z-boundary) -> if this is 1,
     f values for the innermost new z-grid point are left out of dump.dat.
     Note: if EXPAND is 1, ZOFFSET in field.c should be increased by 2
     for the next run.
   wind.c (stvd) -- February 15, 2000
     Modified to use the TVD code, which allows zstep to be different
     from mustep*sstep.  Input zstep.  Modify arguments of stream().
     Call gen_gam().  gam as global variable.
     Variables input from the user:
       starts     Initial value of s = vt (AU)
       stops      Final value of s = vt (AU)
       sstep      s step (AU)
       prints     Printing interval (AU)
       nmu        Number of mu points
       length     Length of simulation region (AU)
       zstep      z step (AU)
       np         Number of momentum points
       p[1..np]   Momenta (energy units)
       m          Particle mass (energy units)
       betasw     v/c of solar wind
       lambda     Scattering mean free path (AU)
       q          Scattering power law index
       printextra  Print extra diagnostic information? (0/1)
   wind.c (s) -- February 1, 1997
     Cosmetic changes in printf() requesting p, m inputs.
   wind.c (s) -- January 27th, 1996
     Add header to "dump.dat" output so that initial() of next stage can
     know nz[w] and nmu.  Also corrected an error in the condition for
     ll in the dump section.
   wind.c -- September 17th, 1995
     Don't call decel() if vsw=0.
   wind.c -- November 12th, 1994
     Passed prints to printout().
   wind.c -- October 16th, 1994
     Changed size of g, h, h3 arrays so can be used in neutrons(),
     printout().  Passed p, betasw to printout().
   wind.c -- August 10th, 1994
     Changed elements() to include eratio[] in deceleration effect.
   wind.c -- April 18th, 1994
     Reset sstep to ensure that zstep=sstep*mustep.
```

```
wind.c -- March 15th, 1994
  Put decel() before stream(). Removed sk[][] and imatrix().
wind.c -- February 27th, 1994
  Added new arrays for stream(), adapted to new format of allocation
  routines in nrutil.c. Imported non-integer ZDUMP mechanism from
  transport.c, modified to start at z=0. New checks on neg, allneg,
  and ntotal. New use of TINY. Corrected errors in mechanism
  for fixing negative fluxes.
wind.c -- May 8th, 1993
  Reworked the variable lprint - set up #define LPRINT.
wind.c -- March 16th, 1993
  Fixed odds and ends. Added dump capability. Added TINY so
  fluxes within TINY of zero become zero, and thus aren't rejected
  if they are negative. Changed TOLERANCE to a combination
  absolute/relative tolerance.
wind.c -- December 12th, 1992
  Modified for the new version of stream(), slide() eliminated,
  corr[w] eliminated. a[w][l][u], etc. are now at
  z=(l-0.5)*zstep again.
wind.c -- November 20th, 1992
  Calls decel() for deceleration.
wind.c -- November 12th, 1992
  elements(), step() changed so that matrix elements are defined
  at 0, zstep, 2*step, ..., length, and a[w][l][u] is the matrix
  element a[u] at z(lab frame) = l*zstep (same for c).
  Now z(lab frame) = (l-1+corr[w])*zstep, so in step() the matrix
  elements are interpolated as (1-corr[w])*a[w][l-1][u]
  + corr[w]*a[w][l][u] (same for c).
wind.c -- November 6th, 1992
  Main program for simulating the transport of solar energetic
  particles. The possible transport processes are:
    streaming
    pitch-angle scattering
    adiabatic focusing
    adiabatic deceleration
    injection
  Key assumptions include neglecting drifts and diffusion
  perpendicular to the magnetic field, and calculating adiabatic
  deceleration as if the solar wind velocity were directed
  along the magnetic field and of a constant magnitude.
  Otherwise, the above processes can be easily modified
  by changing the appropriate subroutines.
  Necessary files and subroutines:
    wind.c     main, elements, step
    decel.c    decel, ci
    field.c    dzz, diffcoeff, antideriv, mudot, arclength, radius,
               cospsi, dsecdz, zenith, decelrate
    initial.c  initial
    inject.c   inject
    nrutil.c   nrerror, dvector, dmatrix, darray,
               free_dvector, free_dmatrix, free_array
    printout.c printout
    stream.c   stream
    tridag.c   tridag
  Variables input from the user:
    starts     Initial value of s = vt (AU)
    stops      Final value of s = vt (AU)
    sstep      s step (AU)
    prints     Printing interval (AU)
    nmu        Number of mu points
```

```
      length     Length of simulation region (AU)
      np         Number of momentum points
      p[1..np]   Momenta (energy units)
      m          Particle mass (energy units)
      betasw     v/c of solar wind
      lambda     Scattering mean free path (AU)
      q          Scattering power law index
      printextra  Print extra diagnostic information? (0/1)
    David Ruffolo and Burin Asavapibhop
    Department of Physics
    Faculty of Science
    Chulalongkorn University
    Bangkok 10330, Thailand
*/
 /*15 May 1992   Montien Tienpratip, Wantana Songprakob, Pisit Lilapattana */
 #include <math.h>
#include <stdio.h>
 #define  C       0.1202
#define  ATOL     1.0e-09  /* Maximum allowable error (absolute)    */
#define  FTOL     1.0e-03  /* Maximum allowable error (fractional)  */
#define  TINY     1.0e-12
#define  LPRINT   500      /* Value of lprint - set to 0 for default */
#define  DUMP     1       /* 0 - DON'T DUMP, 1 - DUMP              */
#define  ZDUMP    2.0     /* Expansion factor for zstep in next run
                           (must be greater than 1).  */
#define  EXPAND   0       /* Expanding inner z-boundary condition?
                           If so, in dump.dat leave out f values
                           for first new z-grid point.          */
 double  ***f, ***gam;
double  ***a, ***c;
double  *aa, *bb, *cc, *an, *bn, *cn, **fs, *g, *h, *h3, *fint, *tosun;
 main()
{
  FILE    *fp_dump;
  double  starts, stops, prints, sstep, nextprint, s;
  double  *beta, *ke, length, m, *p;
  double  aoverv, lambda, q, betasw, vsw, dzz1;
  double  dumpstep, frac, mustep, z, zdump, zstep;
  int     nmu, np, printextra, u, w;
  long    nz, l, ll, maxll;
  double  ***darray(), **dmatrix(), *dvector();
  void    free_darray(), free_dvector(), free_dmatrix();
  void    nrerror();
  double  dzz();
  void    elements(), initial(), printout(), step();
    /* Input parameters from the user */
   printf("Hello! Welcome to wind.\n");
   printf("Please input the following parameters:\n");
   printf("\nStarting value of s=velocity*time (AU): ");
   scanf("%lf",&starts);
   printf("\nFinal value of s (AU): ");
   scanf("%lf",&stops);
   printf("\ns step (AU): ");
   scanf("%lf",&sstep);
   printf("\ns interval after which to print out data (AU): ");
   scanf("%lf",&prints);
   printf("\nNumber of mu points (must be odd): ");
   scanf("%d",&nmu);
   printf("\nLength in the z-direction (AU, best if multiple of sstep): ");
   scanf("%lf",&length);
```

```
printf("\nz step (AU): ");
scanf("%lf",&zstep);
printf("\nNumber of momentum (energy) values: ");
scanf("%d",&np);
 if (np <= 0) nrerror("wind: np <= 0");
 beta  = dvector(1,np);
ke    = dvector(1,np);
p     = dvector(1,np);
tosun = dvector(1,np);
 for (w=1;w<=np;w++) tosun[w] = 0.0;
 printf("\nEnter %d momentum values, from lowest to highest",np);
printf("\n(in MeV/c): ");
for (w=1;w<=np;w++) {
  printf("\n   p[%d] = ",w);
  scanf("%lf",&p[w]);
}
printf("\nEnter the particle mass (in MeV/c^2): ");
scanf("%lf",&m);
printf("\nSolar wind velocity divided by c: ");
scanf("%lf",&betasw);
printf("\nScattering mean free path, lambda (AU): ");
scanf("%lf",&lambda);
printf("\nScattering power-law index, q : ");
scanf("%lf",&q);
printf("\nDo you want me to print extra diagnostic information? ");
printf("\n(enter 1 for yes, 0 for no) ");
scanf("%d",&printextra);
  /* Calculating nz, mustep; length is made an integral
     multiple of zstep.
  */
nz = length / zstep + 0.5;
if (nz == 0) nrerror("wind: nz == 0");
if (printextra) {
  printf("\ninput sstep = %lf, nz(double) = %lf, nz(int) = %ld\n",
   sstep,length/zstep,nz);
}
mustep = 2.0 / (double)nmu;
length = nz * zstep;
if (printextra) {
  printf("\nmustep = %lf, zstep = %lf, new length = %lf\n",
   mustep,zstep,length);
}
  /* Echoing */
printf("You input the following parameters:\n");
printf("\nStarting s (AU)       : %12lf",starts);
printf("\nFinal s (AU)          : %12lf",stops);
printf("\ns step (AU)           : %12lf",sstep);
printf("\nPrint interval (AU)   : %12lf",prints);
printf("\nNumber of mu points   : %5d",nmu);
printf("\nLength*               : %12lf\n",length);
printf("\nz step (AU)           : %12lf\n",zstep);
for (w=1;w<=np;w++) printf("\np[%d] = %12lf",w,p[w]);
printf("\n\nParticle mass        : %12lf",m);
printf("\nSolar wind velocity/c  : %12lf",betasw);
printf("\nLambda (in AU)        : %12lf",lambda);
printf("\nq                     : %12lf",q);
printf("\nPrintextra            : %5d\n",printextra);
printf("\n* reset length to be an integral multiple of zstep\n");
   /* Idiot Proofing */
if (stops < starts) nrerror("wind: stops < starts");
```

```
if (sstep <= 0) nrerror("wind: sstep <= 0");
if (prints < sstep) nrerror("wind: prints < sstep");
if (nmu % 2 == 0 || nmu <= 0) nrerror("wind: nmu is bad");
if (length <= 0) nrerror("wind: length <= 0");
if (zstep <= 0 || zstep >= length) nrerror("wind: bad zstep");
if (p[1] <= 0) nrerror("wind: p[1] <= 0");
for (w=2;w<=np;w++) if (p[w] <= p[w-1]) nrerror("wind: p's reversed");
if (m <= 0) nrerror("wind: m <= 0");
if (betasw < 0 || betasw >= 1) nrerror("wind: betasw is bad");
if (lambda <= 0) nrerror("wind: lambda <= 0");
if (q < 0 || q >= 2) nrerror("wind: q is bad");
 /* Calculating nz, ke, beta, and vsw. */
 for (w=1;w<=np;w++) {
  ke[w] = sqrt(p[w]*p[w]+m*m) - m;
  beta[w] = p[w] / (ke[w]+m);
  if (printextra)
    printf("\nke[%2d] = %12lf,   beta[%2d] = %12lf",w,ke[w],w,beta[w]);
 }
 printf("\n");
 vsw = betasw * C;
   /* Defining arrays */
 f  = darray(1,np,1,nz,1,nmu);
a  = darray(1,np,1,nz,1,nmu);
 c  = darray(1,np,1,nz,1,nmu);
 if (printextra) printf("\n Now we are here step1 \n");
 aa  = dvector(1,nmu);
 bb  = dvector(1,nmu);
 cc  = dvector(1,nmu);
 an  = dvector(1,nmu);
 bn  = dvector(1,nmu);
 cn  = dvector(1,nmu);
 g   = dvector(0,nmu+1>np-1 ? nmu+1 : np-1);
 h   = dvector(0,nmu+1);
 h3  = dvector(0,nmu+1);
 fint = dvector(0,nmu+1);
 fs  = dmatrix(1,nz,1,nmu);
 if (printextra) printf("\n Now we are here step2 \n");
 dzz1 = dzz(mustep,q);
 aoverv = 3.0 * dzz1 / lambda;
 printf("\n q = %lf, dzz = %lf, lambda = %lf, aoverv = %lf\n",q,dzz1,
   lambda,aoverv);
 gen_gam(sstep,beta,m,np,p,nz,zstep,nmu,vsw);
 initial(np,p,nz,zstep,mustep,nmu);
 elements(sstep,beta,np,nz,zstep,mustep,nmu,aoverv,q,vsw,printextra);
 nextprint = starts + prints;
   /* FOR EACH TIME STEP:
       INJECT NEW FLUX (IF NECESSARY).
       CALCULATE f AT THE NEW TIME STEP, ACCORDING TO THE
       TRANSPORT EQUATION.
       PRINT OUT THE DATA (IF NECESSARY).
   */
  for (s=starts;s+sstep<=stops+sstep/2;s+=sstep) {
   inject(s,sstep,beta,np,length,zstep,mustep,printextra);
   step(s,sstep,beta,m,np,p,nz,zstep,nmu,vsw,printextra);
   if (s+sstep >= nextprint-0.01*sstep) {
     printf("\n\n s = %lf\n",s+sstep);
     printout(s+sstep,prints,beta,np,p,length,nz,nmu,betasw);
     nextprint += prints;
   }
  }
```

```
    /* "DUMP" OUT f FOR SELECTED z'S, SO THAT THE RUN MAY BE
CONTINUED.
       Want (ll-0.5) * zdump < nz[w] - 0.5, so the maximum value of ll
       is (nz[w]-0.5)/zdump + 0.5.
     */
   if (DUMP) {
    fp_dump = fopen("dump.dat","w");
     printf("\n\nNow dumping f(mu,z) for a later run...\n");
    zdump = ZDUMP;
    if (zdump < 1.0) zdump = 1.0;
    printf("nz = %ld, zdump = %lf\n",nz,zdump);
    maxll    = (nz-0.5)/zdump + 0.5;
    for (w=1;w<=np;w++) fprintf(fp_dump,"%d  %ld  %d\n",w,maxll,nmu);
    dumpstep = zstep * zdump;
    for (w=1;w<=np;w++) {
      for (ll=1+EXPAND,z=(0.5+EXPAND)*dumpstep;
          ll<=maxll;ll++,z=(ll-0.5)*dumpstep) {
        l   = z/zstep + 0.5;
        frac = z/zstep + 0.5 - (double)l;
        for (u=1;u<=nmu;u++) fprintf(fp_dump,"%12le\n",
          (1.0-frac)*f[w][l][u]+frac*f[w][l+1][u]);
      }
     }
     fclose(fp_dump);
   }
    printf("\nWow! I have finished my work!\n");
    free_dvector(beta,1);
   free_dvector(ke,1);
   free_dvector(p,1);
   free_dvector(tosun,1);
    free_dvector(aa,1);
   free_dvector(bb,1);
   free_dvector(cc,1);
   free_dvector(an,1);
   free_dvector(bn,1);
   free_dvector(cn,1);
   free_dvector(g,0);
   free_dvector(h,0);
   free_dvector(h3,0);
   free_dvector(fint,0);
   free_dmatrix(fs,1,nz,1);
    free_darray(f,1,np,1,nz,1);
   free_darray(a,1,np,1,nz,1);
   free_darray(c,1,np,1,nz,1);
}
 /* elements
     A SUBROUTINE FOR transport.
    CALCULATES ELEMENTS OF MATRICES USED IN step.
*/
 void    elements(sstep,beta,np,nz,zstep,mustep,nmu,aoverv,q,vsw,printextra)
double   sstep, *beta, zstep, mustep, aoverv, q, vsw;
int     np, nmu, printextra;
long    nz;
{
   double   advplus, diffplus, *eratio, mu, muplus, r, v, z;
   int     u, w;
   long    l, lprint;
    double   diffcoeff(), enratio(), mudot(), radius(), cospsi();
   double   *dvector();
   void    free_dvector();
```

```
  eratio = dvector(1,nmu);
  for (w=1;w<=np;w++) {
    v = beta[w] * C;
      /* FIRST, CALCULATE THE SOLAR RADIUS AT EACH POINT. */
    for (l=1;l<=nz;l++) {
      z = zstep*(l-0.5);
      r = radius(z);
        /* CALCULATE MATRICES FOR PITCH-ANGLE SCATTERING AND
FOCUSING.
          IMPLICIT SCATTERING: THE MATRIX IS ONLY NONZERO ALONG
                       THE THREE CENTRAL DIAGONALS.
                       THE (u,u-1) ELEMENT IS a[u].
                       THE (u,u)   ELEMENT IS b[u].
                       THE (u,u+1) ELEMENT IS c[u].
          EXPLICIT SCATTERING: THE MATRIX IS ONLY NONZERO ALONG
                       THE THREE CENTRAL DIAGONALS.
                       THE (u,u-1) ELEMENT IS -a[u].
                       THE (u,u)   ELEMENT IS  d[u] = 2 - b[u].
                       THE (u,u+1) ELEMENT IS -c[u].
        */
        /* IF THERE IS NO PITCH-ANGLE SCATTERING OR FOCUSING, EACH
          ITERATION MATRIX IS THE UNIT MATRIX.
          Set the vector eratio[u] = enratio(mu,z,v,vsw),
          which is needed in the diffusion term.
        */
      for (u=1;u<=nmu;u++) {
        a[w][l][u] = 0.0;
        c[w][l][u] = 0.0;
        mu = -1 + (u-0.5)*mustep;
        eratio[u] = enratio(mu,z,v,vsw);
       }
        muplus  = -1.0 + mustep;
       for (u=1;u<=nmu-1;u++,muplus+=mustep) {
            /* PITCH-ANGLE SCATTERING */
            /* The "4" below appears because each explicit
              or implicit substep accounts for sstep/4.
              After four substeps (explicit and implicit,
              before and after streaming), they account for
              the full sstep.
               During step(), the size of the substep and
              the number of substeps will be repeatedly
              changed by a factor of 2 until the resulting
              f converges to within TOLERANCE.
            */
          diffplus = (sstep/(4*mustep*mustep))
          * diffcoeff(muplus,mustep,aoverv,q,r);
          a[w][l][u+1] -= diffplus * eratio[u];
         c[w][l][u]   -= diffplus * eratio[u+1];
           /* ADIABATIC FOCUSING */
          advplus = (sstep/(8*mustep)) * mudot(v,muplus,r,zstep,vsw);
          a[w][l][u+1] -= advplus;
         c[w][l][u]   += advplus;
        }
      }
    }
      /* IF printextra IS 1, PRINT OUT SELECTED MATRIX ELEMENTS. */
     if (printextra) {
      for (w=1;w<=np;w++) {
        if (w==1 || w==np) {
          printf("\nMatrix Elements: w = %d\n",w);
```

```
        lprint = LPRINT;
        if (lprint <= 0 || lprint >= nz) lprint = nz/4;
        for (l=1;l<=nz;l+=lprint) {
          printf("\n\tl = %ld \n",l);
          for (u=1;u<=nmu;u++)
          printf("a[%3d]=%10lf, c[%3d]=%10lf\n",u,a[w][l][u],u,c[w][l][u]);
          printf("\n");
        }
      }
    }
  }
  free_dvector(eratio,1);
}
/* step
    A SUBROUTINE FOR transport.
    REMINDER OF NOTATION:
      z = ARCLENGTH ALONG MAGNETIC FIELD
      mu = COSINE OF THE PITCH ANGLE
      w = INDEX FOR v-GRID POINTS, FROM 1 TO np
      l = INDEX FOR z-GRID POINTS, FROM 1 TO nz
         z = (l-0.5) * zstep
      u = INDEX FOR mu-GRID POINTS, FROM 1 TO nmu
         mu = (u-0.5) * mustep - 1.0
     i = INDEX FOR mu-GRID POINTS, FROM -(nmu-1)/2 TO +(nmu-1)/2
         mu = i * mustep
    THIS ROUTINE USES THE ALTERNATING DIRECTION IMPLICIT METHOD,
    BUT INSTEAD OF DIFFERENCING IN THE z DIRECTION, EACH GRID
    POINT IS MOVED FROM l TO l + i, WHICH CORRESPONDS TO MOVING
    z TO z + i*zstep = z + mu*sstep (zstep=mustep*sstep).
*/
void    step(s,sstep,beta,m,np,p,nz,zstep,nmu,vsw,printextra)
double  s, sstep, *beta, m, *p, zstep, vsw;
int     np, nmu, printextra;
long    nz;
{
  double  *coarse, *fine, *fl, *swap;
  double  average, asym, betasw, time, timestep;
  int     abcneg, allneg, allzero, bad, loop, n, neg, ntotal, u, w;
  long    l, lprint;
  void    decel(), stream();
  void    nrerror(), tridag();
  lprint = LPRINT;
  if (lprint <= 0 || lprint >= nz) lprint = nz/4;
    /* A NOTE ON POINTER USAGE: THERE ARE 5 ARRAYS USED TO
      STORE f VALUES:
      f[1..np][1..nz][1..nmu] -- MAIN STORAGE OF f(mu,z,v)
      h[1..nmu] ----- AN ESTIMATE OF THE UPDATED f(mu), FOR A FIXED z, v
      h3[1..nmu] ---- "   "   "  "   "    "   " "  "  " "
      g[1..nmu] ----- STORAGE FOR SWAPPING WITH h AND h3
      fint[1..nmu] -- Temporary storage for new estimate, which
                will be rejected if any elements are negative.
      IN ADDITION, FOUR MORE POINTERS HAVE BEEN DEFINED, WHICH
      ARE ASSIGNED TO PARTS OF THE FIXED STORAGE:
      fl = f[w][l] -- POINTS TO f(mu) FOR z = (l-.5)*zstep, v(w)
      coarse -------- POINTS TO THE COARSER ESTIMATE (h OR h3)
      fine ---------- "   "  "  FINER     "     " " "
      swap ---------- FOR SWAPPING coarse AND fine
      aa[u], bb[u], and cc[u] contain matrix elements for a fixed
      z and v.  an[u], bn[u], and cn[u] are the same, but without
      focusing - they are used when focusing leads to negative
```

```
   fluxes.
 */
 /* EXECUTE ONE STEP:
 1) EXPLICIT SCATTERING AND FOCUSING OF f(u) FOR EACH l
      (EXPLICITLY IN mu).
 2) IMPLICIT SCATTERING AND FOCUSING OF f(u) FOR EACH l
      (IMPLICITLY IN mu).
 3) PERFORM THE ABOVE 3 TIMES AS MANY TIMES, FOR ONE THIRD
      THE sstep, USING SMALLER STEPS UNTIL THE DIFFERENCE IS
      BELOW "TOLERANCE".
 4) MOVE f(l,u) TO f(l+i,u).
 5) REPEAT STEPS 1, 2, AND 3.
 */
 if (printextra) printf("\n NUI IN STEP");
for (loop=1;loop<=2;loop++) {
  if (printextra) printf("  loop #%d\n",loop);
  for (w=1;w<=np;w++) {
      /* Perform streaming IN BETWEEN THE
        TWO PITCH-ANGLE DIFFERENCING STEPS.
      */
 if (loop == 2) {
      betasw = vsw / C;
      time   = s / (beta[w]*C);
      timestep = sstep / (beta[w]*C);
      stream(time,timestep,m,np,p,w,nz,zstep,nmu,vsw,printextra);
    }
    for (l=1;l<=nz;l++) {
      fl  = f[w][l];
        /* FIND OUT IF ALL fl[u]'S ARE ZERO (or below TINY). */
     allzero = 1;
      for (u=1;allzero && u<=nmu;u++) {
        if (fl[u] < 0) {
          printf("s = %lf, w = %d, l = %ld, u = %d\n",s,w,l,u);
          nrerror("step: neg # at start of step");
        }
        allzero = fl[u] < TINY;
      }
      if (allzero && printextra && (w==1 || w==np) && l % lprint == 1)
        printf("\nl = %12ld: all entries are 0.0",l);
          /* ONLY PROCEED IF allzero IS FALSE. */
        if (!allzero) {
           /* Initialization for the loop. Note that the ntotal=1
             result is first compared with the current f - if the
             difference is below TOLERANCE, this value is
             accepted.  Otherwise, ntotal is tripled and a and c
             are decreased by a factor of 2 until the result
             converges to within TOLERANCE.  Bomb out if ntotal
             > 10000.
           */
          for (u=1;u<=nmu;u++) {
            aa[u] = a[w][l][u];
            cc[u] = c[w][l][u];
          }
          an[1] = 0.0;
          for (u=1;u<=nmu-1;u++) {
            an[u+1] = (aa[u+1] + cc[u]) / 2.0;
            cn[u]   = (aa[u+1] + cc[u]) / 2.0;
          }
          cn[nmu] = 0.0;
          bb[1]   = 1.0 - aa[2];
```

```
bn[1]  = 1.0 - an[2];
for (u=2;u<=nmu-1;u++) bb[u] = 1.0 - aa[u+1] - cc[u-1];
for (u=2;u<=nmu-1;u++) bn[u] = 1.0 - an[u+1] - cn[u-1];
bb[nmu] = 1.0 - cc[nmu-1];
bn[nmu] = 1.0 - cn[nmu-1];
abcneg  = 1;
fine  = h;
coarse = h3;
for (u=1;u<=nmu;u++) coarse[u] = fl[u];
ntotal = 1;
  /* DO STEPS 1) AND 2) ntotal TIMES. */
 for (;;) {
    /* Check if -an, 2-bn, -cn < 0.  If so,
      immediately jump to increase ntotal.
    */
  if (abcneg) for (u=1,abcneg=0;!abcneg && u<=nmu;u++)
    abcneg = an[u] > 0 || 2.0-bn[u] < 0 || cn[u] > 0;
  if (!abcneg) {
    if (printextra && (w==1 || w==np) && l % lprint == 1)
      printf("\nl=%12ld (%d)",l,ntotal);
    for (u=1;u<=nmu;u++) fine[u] = fl[u];
    for (n=1,allneg=1;n<=ntotal;n++) {
/* EXPLICIT DIFFERENCING IN mu.  fine -> g. */
      g[1] = (2.0-bb[1])*fine[1]-cc[1]*fine[2];
      for (u=2;u<=nmu-1;u++) g[u] = -aa[u]*fine[u-1]
        +(2.0-bb[u])*fine[u]-cc[u]*fine[u+1];
      g[nmu] = -aa[nmu]*fine[nmu-1]+(2.0-bb[nmu])*fine[nmu];
        /* IMPLICIT DIFFERENCING IN mu.
tridag SOLVES THE MATRIX EQUATION M(i,j) x(j) =
           y(i), FOR A TRIDIAGONAL M.  THE FIRST 3 ARGUMENTS
           POINT TO MATRIX ELEMENTS (TO M(u,u-1)'S,
           M(u,u)'S, AND M(u,u+1)'S), THE FOURTH ARGUMENT
           POINTS TO y, THE FIFTH POINTS TO x, AND THE LAST
           IS THE DIMENSION OF THE MATRICES AND VECTORS.
               g -> fint.
         */
          tridag(aa,bb,cc,g,fint,nmu);
         /* Set fluxes between -TINY and zero to zero.
           Check for negative numbers.  If they appear,
           use matrix elements with the effect of focusing
           removed (an[u], bn[u], and cn[u]).  Bomb out
           if negative numbers persist.  Do not allow
           the iteration to end if each substep yielded
           negative numbers.
         */
        for (u=1;u<=nmu;u++)
          if (-TINY < fint[u] && fint[u] < 0) fint[u] = 0.0;
        for (u=1,neg=0;!neg && u<=nmu;u++) neg = fint[u] < 0.0;
        if (!neg) allneg=0;
        if (printextra && (w==1 || w==np) && l%lprint==1
          && neg) printf(" -");
        if (printextra && (w==1 || w==np) && l%lprint==1
          && !neg) printf(" +");
        if (!neg) {
          for (u=1;u<=nmu;u++) fine[u] = fint[u];
        } else {
          g[1] = (2.0-bn[1])*fine[1]-cn[1]*fine[2];
          for (u=2;u<=nmu-1;u++) g[u] = -an[u]*fine[u-1]
            +(2.0-bn[u])*fine[u]-cn[u]*fine[u+1];
          g[nmu] = -an[nmu]*fine[nmu-1]
```

```
                    +(2.0-bn[nmu])*fine[nmu];
                  tridag(an,bn,cn,g,fine,nmu);
                  for (u=1,neg=0;!neg && u<=nmu;u++)
                    neg = fine[u] < 0.0;
                  if (neg) {
                    printf("s = %lf, w = %d, l = %ld\n",s,w,l);
                    nrerror("step: neg after special treatment");
                  }
                }
              }
              average = 0.0;
              for (u=1;u<=nmu;u++) average += fine[u];
              average /= nmu;
              for (u=1,bad=0;!bad && u<=nmu;u++)
                bad = fabs(coarse[u] - fine[u]) > ATOL
                  && fabs(coarse[u] - fine[u]) > FTOL*average;
              if (!bad && !allneg) break;
               swap   = fine;
              fine   = coarse;
              coarse = swap;
            }
            ntotal *= 2;
            if (ntotal > 10000) {
              printf("Bombing out! ");
              printf("s=%lf, w=%d, l=%ld\n",s,w,l);
              for (u=1;u<=nmu;u++) printf("fl[%d]=%lf\n",u,fl[u]);
              nrerror("step: ntotal > 10000");
            }
            for (u=1;u<=nmu;u++) {
              aa[u] /= 2.0;
              an[u] /= 2.0;
              bb[u]  = 1.0 + (bb[u] - 1.0) / 2.0;
              bn[u]  = 1.0 + (bn[u] - 1.0) / 2.0;
              cc[u] /= 2.0;
              cn[u] /= 2.0;
            }
          }
            /* MOVE BEST ESTIMATE, fine[u], TO fl[u] */
        for (u=1;u<=nmu;u++) fl[u] = fine[u];
        }
      }
    }
      /* Deceleration for all w values at once, in between the
        two pitch-angle differencing steps.
      */
    if (loop == 1 && vsw > 0) {
      decel(s,sstep,m,np,p,lprint,nz,zstep,nmu,vsw,printextra);
    }
  }
}
```

Vitae

Name: Miss Thiranee Khumlumlert

Permanent Position: Faculty member, Department of Physics (since 1997)

Permanent Address: Department of Physics

Faculty of Science

Naresuan University

Phitsanulok, Thailand

Born: December 27^{th}, 1970 in Bangkok, Thailand

Education: 1992 B.Sc. (2nd Class Honors) degree in Physics

Naresuan University, Thailand

1996 M.Sc. degree in Physics

Chulalongkorn University, Thailand