GALPROP *Explanatory Supplement*

Andrew W. Strong*

Max-Planck-Institut für extraterrestrische Physik,
Postfach 1312, 85741 Garching, Germany

Igor V. Moskalenko†

Hansen Experimental Physics Laboratory,
Stanford University, Stanford, CA 94305, U.S.A.

Troy A. Porter‡

Hansen Experimental Physics Laboratory,
Stanford University, Stanford, CA 94305, U.S.A.

Gudlaugur Jóhannesson§

Science Institute, University of Iceland Dunhaga 5 IS-107 Reykjavk, Iceland

Elena Orlando¶

Hansen Experimental Physics Laboratory,
Stanford University, Stanford, CA 94305, U.S.A.

Seth W. Digel ‖

SLAC National Accelerator Laboratory,
2575 Sand Hill Road,
Menlo Park, CA 94025, USA

Andrey E. Vladimirov**

Hansen Experimental Physics Laboratory,
Stanford University, Stanford, CA 94305, U.S.A.

April 11, 2013

Abstract

This document provides a description of the *GALPROP* cosmic ray propagation code.

*e-mail: `aws@mpe.mpg.de`
†e-mail: `imos@stanford.edu`
‡e-mail: `tporter@stanford.edu`
§e-mail: `gudlaugu@glast2.stanford.edu`
¶e-mail: `eorlando@stanford.edu`
‖e-mail: `digel@stanford.edu`
**e-mail: `avladim@stanford.edu`

Contents

1 Introduction

The origin of cosmic rays have been intriguing scientists since 1912 when V. Hess carried out his famous balloon flight to measure the ionisation rate in the upper atmosphere. The cosmic rays are energetic particles, which come to us from outer space, and are measured either through satellites, balloons, or Earth based experiments. The spectrum of cosmic rays can be approximately described by a single power law with index -3 from ~ 10 GeV to the highest energies ever observed $\sim 10^{20}$ eV. The only feature observed below 10^{18} eV is a knee around 10^{15} eV. Because of this featureless spectrum, it is believed that cosmic-ray production and propagation is governed by the same mechanism over decades of energy, the same mechanism at least works below the knee and the same or another one works above the knee. Meanwhile the origin of the cosmic rays spectrum is not still understood.

Galactic cosmic rays are important part of the interstellar medium. The energy density of relativistic particles is about 1 eV cm^{-3} and is comparable to the energy density of interstellar radiation field, magnetic field, and turbulent motions of the interstellar gas. This makes cosmic rays one of the essential factors determining the dynamics and processes in the interstellar medium.

The sources of cosmic rays are believed to be supernovae and supernova remnants, pulsars, compact objects in close binary systems, and stellar winds. Observations of X-ray and γ-ray emission from these objects reveal the presence of energetic particles thus testifying to efficient acceleration processes near these objects. Particles accelerated near the sources propagate tens of millions years in the interstellar medium where they lose or gain energy, their initial spectra and composition change, they produce secondary particles and γ-rays. The distruction of primary nuclei via spallation gives rise to secondary nuclei and isotopes which are rare in nature, antiprotons, and charged pions that decay producing secondary positrons and electrons.

The variety of isotopes in cosmic rays allows one to study different aspects of their acceleration and propagation in the interstellar medium as well as the source composition. Stable secondary nuclei tell us about the diffusion coefficient and Galactic winds (convection) and/or re-acceleration in the interstellar medium (2nd order Fermi acceleration mechanism). Long-lived radioactive secondaries allow one to constrain global Galactic properties such as Galactic halo size. Abundances of K-capture isotopes, which being stopped in the interstellar gas would decay via electron K-capture, allow one to probe the gas density and acceleration time scale. All these together allow us in principle to build a model of particle acceleration and propagation in the Galaxy.

Such a model is however incomplete. The whole of our knowledge is based on measurements done only at one point on the outskirts of the Galaxy, the solar system, and the assumption that particle spectra and composition are (almost) the same at every point of the Galaxy. The latter may not necessarily be correct. γ-rays are able to deliver the information directly from distant regions thus complementing that obtained from cosmic-ray measurements. Some part of the diffuse γ-rays is produced in energetic nucleons interactions with gas via neutral pion production, another is produced by electrons via inverse Compton scattering and bremsstrahlung. These processes are dominant in different parts of the spectra of γ-rays, therefore, if deciphered the γ-ray spectrum can provide information about the large-scale spectra of nucleonic and leptonic components of cosmic rays.

To extract information which is contained in cosmic ray abundances and γ-ray fluxes one needs to develop a model of particle production and propagation in the Galaxy. Though the basic features of particle diffusion in the Galaxy seem to be well-established, the continuous flow of new more and more accurate data from space, balloon and ground based experiments motivates further development of models. Analytical and semi-analytical models are able to interpret one or only a few features and often fail when they try to deal with the whole variety of data. Therefore more realistic and consistent models are required which would be able to incorporate many processes and astrophysical data of many different kinds simultaneously: nuclear reaction networks and nuclear cross sections, production of antiprotons, positrons, γ-rays and synchrotron emission, realistic gas distribution, radiation field distribution and spectrum, energy losses, convection, diffusive re-acceleration etc.

In several years new missions planned for cosmic ray experiments will tremendously increase the quality and accuracy of cosmic-ray data making new progress impossible without highly developed models. Data will continue to flow from the high resolution detectors on Ulysses, Advanced Composition Explorer and Voyager space missions. During the next few years there will be several flights of ballon-borne high resolution spectrometers that will extend our knowledge of antiproton, positron and electron spectra in cosmic rays. Several more high resolution space experiments are planned to be launched in the next 2–3 years, e.g., PAMELA to measure antiprotons, positrons, electrons, and isotopes H through C over the energy range of 0.1 to 200 GeV, the Alpha

Magnetic Spectrometer will measure particle and nuclear spectra to TeV energies. The previous γ-ray mission EGRET, one of the four detectors on board of the Compton Gamma-Ray Observatory, gave a detailed map of the Galactic diffuse emission in the range 30 MeV – 10 GeV which traces the cosmic ray distribution in the Galaxy and possibly the acceleration sites of cosmic rays. The current Fermi-LAT mission capability covers the range 30 MeV – 1 TeV with a sensitivity two orders of magnitude better.

Clearly, a detailed model of cosmic ray propagation in the Galaxy should supplement the high quality data obtained by the spacecraft and balloon-borne missions, providing support for the necessary interpretation and analysis.

Recent developments of *galprop* with corresponding results can be found in Strong et al. (2004a,b,c). A review of subject and the context and philosophy of *galprop* can be found in Strong et al. (2007). A short paper summarizing this release v54 is in Strong et al. (2009). A description of the enhancements and Web interface is in Vladimirov et al. (2010).

1.1 Versions of *GALPROP* and its Explanatory Supplement

In this manual we describe *galprop* version 54, released in September 2010, and which succeeds version 50p. It replaces the previous manual which was called galprop_v50.ps.

This manual is still being updated, and is not completely up-to-date for v54. All input parameters are described, but the code documentation and description of some features is still incomplete. New versions will be issued as available. Up-to-date details on compiling and running GALPROP are given in the README file from the GALPROP distribution.

Comments, questions, errors found, and suggestions are welcome and should be addressed to the authors.

2 General Principles

2.1 Transport equation

GALPROP solves the transport equation with a given source distribution and boundary conditions for all cosmic-ray species. This includes Galactic wind (convection), diffusive reacceleration in the interstellar medium, energy losses, nuclear fragmentation, and decay. The numerical solution of the transport equation is based on an implicit second-order scheme (Press et al., 1992). The spatial boundary conditions assume free particle escape. Since we have a 3-dimensional (R, z, p) or 4-dimentional (x, y, z, p) problem (spatial variables plus momentum) we use "operator splitting" to handle the implicit solution.

The propagation equation is written in the form:

$$\frac{\partial \psi}{\partial t} = q(\vec{r}, p) + \vec{\nabla} \cdot (D_{xx}\vec{\nabla}\psi - \vec{V}\psi) + \frac{\partial}{\partial p}\, p^2 D_{pp} \frac{\partial}{\partial p} \frac{1}{p^2}\, \psi - \frac{\partial}{\partial p}\left[\dot{p}\psi - \frac{p}{3}\,(\vec{\nabla}\cdot\vec{V})\psi\right] - \frac{1}{\tau_f}\psi - \frac{1}{\tau_r}\psi\ , \tag{1}$$

where $\psi = \psi(\vec{r}, p, t)$ is the density per unit of total particle momentum, $\psi(p)dp = 4\pi p^2 f(\vec{p})$ in terms of phase-space density $f(\vec{p})$, $q(\vec{r}, p)$ is the source term, D_{xx} is the spatial diffusion coefficient, $\vec{V}$ is the convection velocity, reacceleration is described as diffusion in momentum space and is determined by the coefficient D_{pp}, $\dot{p} \equiv dp/dt$ is the momentum loss rate, τ_f is the time scale for fragmentation, and τ_r is the time scale for the radioactive decay. The details of the numerical scheme is described in § 3.

For a given halo size the diffusion coefficient as a function of momentum and the reacceleration or convection parameters is determined by boron-to-carbon ratio data. The spatial diffusion coefficient is taken as $D_{xx} = \beta D_0(\rho/\rho_0)^\delta$ if necessary with a break ($\delta = \delta_{1,2}$ below/above rigidity ρ_0), where the factor β ($= v/c$) is a consequence of a random-walk process. For the case of reacceleration the momentum-space diffusion coefficient D_{pp} is related to the spatial coefficient D_{xx} (Berezinskii et al., 1990; Seo & Ptuskin, 1994), where $\delta = 1/3$ for a Kolmogorov spectrum of interstellar turbulences. The convection velocity (in z-direction only) $V(z)$ is assumed to increase linearly with distance from the plane ($dV/dz > 0$ for all z); this implies a constant adiabatic energy loss. The linear form for $V(z)$ is consistent with cosmic-ray driven MHD wind models (Zirakashvili et al., 1996). Since the wind cannot blow in both directions at $z = 0$ this formulation requires a zero velocity there. A more general case where the wind starts at $z = \pm z_0$ and is zero for $|z| < z_0$ has therefore been implemented; in this case $dV/dz = 0$ will give a constant wind velocity equal to the value at z_0.

The distribution of cosmic-ray sources (Strong & Moskalenko, 1998) is chosen to reproduce the cosmic-ray distribution determined by analysis of EGRET γ-ray data (Strong & Mattox, 1996). The injection spectrum of nucleons is assumed to be a power law in momentum, $dq(p)/dp \propto p^{-\gamma}$. Energy losses (Strong & Moskalenko, 1998) for nucleons by ionization and Coulomb interactions are included, and for electrons by ionization, Coulomb interactions, bremsstrahlung, inverse Compton, and synchrotron. The total magnetic field distribution is adjusted to match the 408 MHz synchrotron longitude and latitude distributions. This is in agreement with interstellar field estimates (Broadbent et al., 1990) and other magnetic field models (e.g., Heiles, 1996; Vallée, 1996).

2.1.1 Interstellar hydrogen distribution

The interstellar hydrogen distribution uses H I and CO surveys and information on the ionized component (Moskalenko et al., 2001b); the helium fraction of the gas is taken as 0.11 by number. The H_2 gas number density is defined in the form of table (Bronfman et al., 1988), which is interpolated linearly, and the conversion factor is taken as $X \equiv n_{\mathrm{H}_2}/\epsilon_{\mathrm{CO}} = 1.9 \times 10^{20}$ mols. cm^{-2}/(K km s^{-1}) (Strong & Mattox, 1996). The H I gas number density in the Galactic plane is defined by a table (Gordon & Burton, 1976) which is renormalized to agree with the total integral perpendicular to the plane by Dickey & Lockman (1990). The z-dependence is calculated using the approximation by Dickey & Lockman (1990) for $R < 8$ kpc, using the approximation by Cox et al. (1986) for $R > 10$ kpc, and interpolated in between. The ionized component H II (atom cm^{-3}) is calculated using a cylindrically symmetrical model (Cordes et al., 1991).

2.1.2 Interstellar radiation field (ISRF)

For calculation of the spectrum of γ-rays arising from inverse Compton scattering and electron energy losses, the full ISRF as function of (R, z, ν) is required, which was previously not available in the literature. Our ISRF calculation uses emissivities based on stellar populations and dust emission. The infrared emissivities per atom of H I and H_2 are based on COBE/DIRBE data from Sodrowski et al. (1997), combined with the distribution of H I and H_2. The spectral shape is based on the silicate, graphite and PAH synthetic spectrum using COBE data from Dwek et al. (1997). For the distribution of the old stellar disk component we use the model of Freudenreich (1998) based on the COBE/DIRBE few micron survey. The stellar luminosity function is taken from Wainscoat et al. (1992). For each stellar class the local density and absolute magnitude in standard optical and near-infrared bands is given, and these are used to compute the local stellar emissivity by interpolation in wavelength. The z-scaleheight for each class and the spatial functions (disk, halo, rings, arms) given by Wainscoat et al. (1992) then give the volume emissivity as a function of position and wavelength. All their main-sequence and AGB types were explicitly included.

A new model of the ISRF is now available, see Porter &Strong (2005); Moskalenko et al. (2006).

2.1.3 Gamma rays

Gas-related γ-ray intensities are computed from the emissivities as a function of (R, z, E_γ) using the column densities of H I and H_2 for Galactocentric annuli based on 21-cm and CO surveys. Neutral pion production is calculated using a formalism by Dermer (1986a,b) as described in Moskalenko & Strong (1998); bremsstrahlung is calculated using a formalism by Koch & Motz (1959) as described in Strong et al. (2000). The inverse Compton scattering is treated using the formalism for an isotropic or anisotropic radiation field developed by Moskalenko & Strong (2000a), and this uses the interstellar radiation field calculations as described above.

NB in v54 the anisotropic calculation is not yet implemented.

2.1.4 Nuclei H to Ni

In the new version, the code is updated to include the cross-section measurements and energy dependent fitting functions (Strong & Moskalenko, 2001). The nuclear reaction network is built using the Nuclear Data Sheets. Currently, the isotopic cross section database consists of more than 2000 points collected from sources published in 1969–1999. This includes a critical re-evaluation of some data and cross checks. The isotopic cross sections are calculated using the author's fits to major beryllium and boron production cross sections. Other cross sections are calculated using phenomenological approximations by Webber et al. (1990) (code `WNEWTR.FOR` version of

1993) and/or Silberberg and Tsao (code `YIELDX_011000.FOR` version of 2000) renormalized to the data where it exists. The cross sections on the He target are calculated using a parametrization by Ferrando et al. (1988). For pp and pA inelastic cross sections we adapted parametrizations by Tan & Ng (1983a) and Letaw et al. (1983). The reaction network is solved starting at the heaviest nuclei (i.e., ^{64}Ni). The propagation equation is solved, computing all the resulting secondary source functions, and then proceeds to the nuclei with $A-1$. The procedure is repeated down to $A=1$. Our preliminary results for all cosmic ray species $Z \leq 28$ are given in Strong & Moskalenko (2001) and re-evaluation of the radioactive isotopes of Be, Al, Cl, Mn is given in Moskalenko et al. (2001a).

2.1.5 Secondary antiprotons

The code calculates production and propagation of secondary antiprotons as described in Moskalenko et al. (1998) and Moskalenko et al. (2001b). Antiproton production in pp-collisions has been calculated using the parametrization of the invariant $\bar{p}$-production cross section given by Tan & Ng (1983b). The antiproton production by nuclei with $Z \geq 2$ is calculated using effective nuclear factors; the latter computed using the Monte Carlo event generator DTUNUC (Roesler et al., 1998; Simon et al., 1998) or scaling factors similar to Gaisser & Schaefer (1992). For the inelastic proton cross sections we adapted parametrizations by Tan & Ng (1983a) and Letaw et al. (1983). The total $\bar{p}p$ inelastic cross section has been calculated using a fit from Tan & Ng (1983a) and parametrization by Groom et al. (2000). The antiproton absorption cross section on nuclear targets is calculated following Moiseev & Ormes (1997). Inelastically scattered antiprotons are treated as a separate "tertiary" component.

2.1.6 Secondary positrons and electrons

Secondary positrons and electrons in cosmic rays are the final product of decay of charged pions and kaons which in turn created in collisions of cosmic-ray particles with gas. Pion production in pp-collisions is considered following a method developed by Dermer (1986a,b), which combines isobaric (Stecker, 1970) and scaling (Badhwar et al., 1977; Stephens & Badhwar, 1981) models of the reaction. Secondary positron and electron production is computed as described in Moskalenko & Strong (1998), that includes a critical reevaluation of the charged pion and kaon decay calculations. Primary electrons are computed in the same propagation model.

2.2 Working quantities

The nuclei are aligned on the same kinetic energy per nucleon E_{kin} since this simplifies the secondary-to-primary computation, where primaries produce secondaries of the same E_{kin}. However the basic CR density used has *units* of density per total momentum p since this is natural for propagation. The actual units used internally are $\frac{c}{4\pi}n(p)$, where $n(p) = dn/dp$ in units of cm^{-3} MeV^{-1}.

When the *flux* $I(E_{kin})$ in cm^{-2} sr^{-1} s^{-1} (MeV/nucleon)$^{-1}$ is necessary, it can be simply obtained from

$$I(E_{kin}) = \frac{\beta c}{4\pi}\frac{dn}{dp}\frac{dp}{dE_{kin}} = \frac{c}{4\pi}n(p)A, \tag{2}$$

where A is the nucleus mass number. This follows from $dp = \frac{A}{\beta}dE_{kin}$. The combined requirements of transport and fragmentation are thus elegantly met. The normal units for presentation of CR data are cm^{-2} sr^{-1} s^{-1} (MeV/nucleon)$^{-1}$, and with this scheme the conversion is trivial. The nucleus energy scales are logarithmic in E_{kin}.

The *output* nuclei spectra are fluxes[1] as defined above, multiplied by E^2_{kin}. The spectra are at $z=0$ as a function of R in kpc. The flux spectra are directly comparable with experimental data at $R = R_0$.

2.3 Abundances and normalization

All calculations are done treating $n(p)$ as the basic quantity; the final step is to normalize to the absolute proton or electron fluxes given in the *galdef* file. All other primary and secondary nuclei follow the same normalization factor as for protons since source abundances relative to protons are specified in the *galdef* file. The global

[1] Note this differs from the old f90 version which outputted $\frac{c}{4\pi}p^2 n(p)$.

normalization to the absolute proton flux is applied at the end of the entire propagation, in `nuclei_normalize`. Only for output is the *flux* computed for all nuclei, in routine `store_gcr`. The π^0-decay emissivities depend on p, He and the bremsstrahlung and IC emissivities depend on electrons. The nuclei and electron normalizations are therefore done before computing gammas and synchrotron.

2.4 Secondary production

For computation of gamma-ray emissivities and source functions for secondary positrons, electrons and antiprotons, the integral over nucleon energies is required.

The nucleus energy scales are uniform in $\log E_{kin}$. It is easy therefore to replace integration over kinematic variable with summation over $\Delta(\log E_{kin})$. In case of secondary source function for, e.g., antiprotons it can be calculated as following

$$q(p) = \beta c n_H \int dp' \frac{d\sigma(p,p')}{dp} n(p'), \tag{3}$$

where n_H is the gas density (in this case pure Hydrogen), $d\sigma(p,p')/dp$ is the production cross section, $n(p')$ is the CR proton *density*, and p' is the total momentum of a nucleus. Substitution of dp' with $d(\log E_{kin})$ gives:

$$\begin{aligned} q(p) &= c n_H A \int d(\log E_{kin})\, E_{kin}\, n(E_{kin}) \frac{d\sigma(p,E_{kin})}{dp} \\ &= c n_H A\, \Delta(\log E_{kin}) \sum_{E_{kin}} E_{kin}\, n(E_{kin}) \frac{d\sigma(p,E_{kin})}{dp}, \end{aligned} \tag{4}$$

where we used $dp' = \frac{1}{\beta} A E_{kin} d(\log E_{kin})$.

2.5 Coordinate Systems

under construction

`galprop` works either in 2D (R, z) or 3D (x, y, z) coordinates; these are consistently defined, with $R = \sqrt{(x^2 + y^2)}$. The system is chosen for simplicity and after reviewing the literature [2]. Here put a diagram !

In `galprop` we adopt a system with the Galactic centre at (0,0,0) and the Solar position on the +ve x axis at $x = R_o$. The system is right-handed, i.e. $Z = X \times Y$, so that $x = R_o - s\cos(b)\cos(l), y = -s\cos(b)\sin(l), z = s\sin(b)$, where s is the distance from the Sun. In the RH system, Looking down from the +Z axis, x is to the right and y is up.

NB at present the B-field sometimes uses a LH system ($Z = -X \times Y$) for consistency with some recent usages (to be clarified). In the LH system, looking down from the +Z axis, x is to the right and y is down, $y = s\cos(b)\sin(l)$. The parity only has importance at present for the B-field, but would be relevant when e.g. spiral structure is also included in the cosmic-ray source function. The Sun et al. 2008 model uses a RH system like `galprop`, but with the solar position on the -ve x-axis unlike `galprop` (+ve x-axis).

The 3D structure of the gas for making gamma-ray maps is implemented via observed column-density maps in (l,b), so is independent of these definitions. A more detailed description including pitch angles for spirals is given in the section on B-fields and synchrotron *(to be written)*.

An arbitrary camera location is now possible in `galprop`. The camera location is given as an (x,y,z) triple and the skymap outputs are such that l,b = 0,0 always points along the negative x-axis, l,b = 90°,0 points along the negative y-axis, and l,b = 0,90° points in the positive z-axis. This may change in the future with an option to rotate the field of view.

[2] There appears to be no accepted standard convention (even from the IAU) for the orientation and parity of the 3D system, and the convention adopted is just one of those in common use. Some have the Sun on the +ve x-axis (e.g. Drimmel and Spergel 2001, LH system), on the -ve x-axis (e.g. Sun etal 2009, Jansson etal 2009, RH system), others on the +ve y-axis (e.g. Taylor and Cordes 1993, NE2001 electron density model of Cordes and Lazio http://arxiv.org/abs/astro-ph/0207156), RH system. Han and Qiao 1994 do not explicitly use x,y but their system is equivalent to Sun on +x, LH system, since the azimuth angle θ increases clockwise.

3 The galprop Code

galprop is the main program which calls the routines to read the *galdef* file, read the nuclear data arrays, fill in the gas and radiation field distributions in the Galaxy, create cosmic rays, propagate particles, and finally store the output arrays.

The code consists of C++ source, with a few (for historical reasons) FORTRAN 77 files. The code distribution is a gzipped tar file. The executable galprop is built automatically using the autotools package. Data files for cross-sections are also part of the package.

It requires several public-domain external packages: cfitsio, CCfits, healpix, GSL, CLHEP. Details of how to build galprop are given in the README file.

Astronomical data required by galprop (HI, CO surveys, interstellar radiation field) are provided as separate datasets in a separate distribution.

Below are listed the routines' names with a short description.

3.1 C++ files

This is not complete for v54: to be updated.

3.1.1 Headers

Configure.h	Galdef.h	constants.h	galprop_classes.h
Distribution.h	Particle.h	fort_interface.h	global.h
Galaxy.h	Spectrum.h	galprop.h	

3.1.2 Routines

B_field_model.cc
Configure.cc
D_pp.cc
Distribution.cc
Galaxy.cc
Galdef.cc
He_to_H_CS.cc
IC_anisotropy_factor.cc
IC_cross_section.cc
Particle.cc
create_SNR.cc
create_galaxy.cc
create_gcr.cc
create_transport_arrays.cc
decayed_cross_sections.cc
e_KN_loss.cc
electrons_normalize.cc
energy_losses.cc
fort_interface1.cc
fort_interface2.cc
galprop.cc
gen_IC_emiss.cc
gen_IC_skymap.cc
gen_bremss_emiss.cc
gen_bremss_ionized_skymap.cc
gen_ionization_rate.cc
gen_isrf_energy_density.cc
gen_pi0_decay_emiss.cc
ionization_bethe.cc
isrf_energy_density.cc
kinematic.cc
nH2.cc
nHI.cc
nHII.cc
nuc_package.cc
nuclei_normalize.cc
print_BC.cc
propagate_particles.cc
propel.cc
propel_diagnostics.cc
read_isrf.cc
sigma_boron_dec_heinbach_simon.cc
source_SNR_event.cc
source_distribution.cc
store_IC_skymap.cc
store_IC_skymap_comp.cc
store_bremss_emiss.cc
store_bremss_ionized_skymap.cc
store_gcr.cc
store_gcr_full.cc
store_ionization_rate.cc
store_pi0_decay_emiss.cc
store_synch_skymap.cc
test_Distribution.cc
test_Particle.cc
test_float_accuracy.cc

```
gen_secondary_antiproton_source.cc  test_isotope_cs.cc
gen_secondary_positron_source.cc    test_nH.cc
gen_secondary_proton_source.cc      test_source_SNR_event.cc
gen_secondary_source.cc             test_suite.cc
gen_synch_emiss.cc                  tridag.cc
gen_synch_skymap.cc                 tridag_double.cc
gen_tertiary_antiproton_source.cc   tridag_sym.cc
global.cc
```

3.2 FORTRAN files

```
WNEWTR_FUNC_aws.f     bremss_spec.f     inter.f       synchrotron.f
YIELDX_011000_imos.f  cfactor.f         nucleon_cs.f
antiproton.f          e_loss_compton.f  pp_meson.f
```

3.3 Input data files:

Nuclear physics:

`WNEWTR_082693.CDR  isotope_cs.dat  nucdata.dat  p_cs_fits.dat`

`barpol.dat eval_iso_cs.dat`

Galactic structure:

`rbands_hi7.fits.gz:`

.... HI surveys in 9 Galactocentric rings

`rbands_co4.fits.gz:`

.... CO surveys in 9 Galactocentric rings

`MilkyWay_DR0.5_DZ0.1_DPHI10_RMAX20_ZMAX5_galprop_format.fits`

....ISRF interstellar radiation field as function of (R, z, λ)

3.4 Output data files

All data output is entirely in the form of FITS files. The names are appended with the ID of the galdef file used to generate them, in this example

`galdef_50_600203a`

was used so they are appended with

`50_600203a`

This allows unique identification of the version and parameters used to produce the results. The tag, in our example 600203a, can be any (no blanks) string up to 10 characters long. (Our publications include the ID on the tables/plots for easy reference and we encourage this practice.)

```
nuclei_50_600203a
nuclei_full_50_600203a
```

Spectrum of all nuclei, electrons and positrons, as function of kinetic energy.

Units: (MeV/nucleon)^2 cm^{-2} sr^{-1} s^{-1} (MeV/nucleon)$^{-1}$.

First form has spectra at z=0 only, for 2D as function of (R, E,species), for 3D as function of (x, y, E,species)[3]. Second form is full spatial 2D (function of R, z, E,species) or spatial 3D (function of x, y, z, E,species) and used e.g. for a warm start as described for the corresponding parameter in Section 4. Header contains keywords identifying the particles and the (logarithmic) energy scale. The particles are ordered by Z, A, K (K=K electron). In the ambiguous case of $Z = -1, A = 0$ which can be primary or secondary electrons, the secondaries are stored first in case both are present. [4]

Example of nuclei 2D file FITS header:

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -32 / number of bits per data pixel
NAXIS   =                    4 / number of data axes
NAXIS1  =                   21 / length of data axis 1
NAXIS2  =                    1 / length of data axis 2
NAXIS3  =                   11 / length of data axis 3
NAXIS4  =                   21 / length of data axis 4
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
CRVAL1  =                   0. / Start of axis 1           R (kpc) =      CRVAL1 + i * CDELT1
CRVAL2  =                   0. / Start of axis 2           not used
CRVAL3  =                   3. / Start of axis 3           log10(E/MeV) = CRVAL3 + k * CDELT3
CRVAL4  =                   1. / Start of axis 4           sequential species number
CDELT1  =                   1. / Increment of axis 1
CDELT2  =                  0.1 / Increment of axis 2
CDELT3  =    0.301029995663981 / Increment of axis 3
CDELT4  =                   1. / Increment of axis 4
NUCZ001 =                    1 / Z of nucleus 1                          secondary positrons
NUCA001 =                    0 / A of nucleus 1
NUCK001 =                    0 / K-electrons of nucleus 1
NUCZ002 =                   -1 / Z of nucleus 2                          secondary electrons
NUCA002 =                    0 / A of nucleus 2
NUCK002 =                    0 / K-electrons of nucleus 2
NUCZ003 =                   -1 / Z of nucleus 3                          primary   electronrs
NUCA003 =                    0 / A of nucleus 3
NUCK003 =                    0 / K-electrons of nucleus 3
NUCZ004 =                    1 / Z of nucleus 4                          protons
NUCA004 =                    1 / A of nucleus 4
NUCK004 =                    0 / K-electrons of nucleus 4
NUCZ005 =                    1 / Z of nucleus 5                          deuterium
NUCA005 =                    2 / A of nucleus 5
NUCK005 =                    0 / K-electrons of nucleus 5
NUCZ006 =                    2 / Z of nucleus 6                          3He
NUCA006 =                    3 / A of nucleus 6
NUCK006 =                    0 / K-electrons of nucleus 6
NUCZ007 =                    2 / Z of nucleus 7                          4He
NUCA007 =                    4 / A of nucleus 7
NUCK007 =                    0 / K-electrons of nucleus 7
```

The nuclei 3D file (nuclei_full) has additional keywords

[3] to be checked for 3D!

[4] In future primary positrons will be included, in which case the same principle holds: the secondaries are stored first.

```
ELENORM = 3.77123467891234E-34 / Electron norm factor
NUCNORM = 1.85123456718234E-25 / Proton norm factor
```

These are used when a warm start is made, since the file is read in and needs to be rescaled to the internal GALPROP units before normalization to the GALDEF normalization values. They are also useful when the user would like to compare fluxes independent of the normalization, for example to investigate the effect of reacceleration on the absolute spectrum for the same source input. They are defined so that normalized flux = internal flux $\times$ ELENORM or NUCNORM. Hence to get unnormalized fluxes, divide the flux in the file by ELENORM or NUCNORM.

```
bremss_emiss_50_600203a
bremss_skymap_50_600203a
bremss_HIR_skymap_50_600203a
bremss_ionized_skymap_50_600203a
```

Bremsstrahlung emissivities(R, z, E_γ) and skymaps(l, b, E_γ) as function of γ-ray energy. First skymap is integrated over full line-of-sight. HIR, H2R signify separated contributions from HI and H_2, each with Galactocentric rings. ionized indicated contribution from HII gas.

Units: emissivity: MeV^2(H atom)$^{-1}$ sr^{-1} s^{-1} MeV^{-1}, skymaps: MeV^2 cm^{-2} sr^{-1} s^{-1} MeV^{-1}

```
pion_decay_emiss_50_600203a
pion_decay_skymap_50_600203a
pion_decay_HIR_skymap_50_600203a
pion_decay_H2R_skymap_50_600203a
```

Pion-decay emissivities(R, z, E_γ) and skymaps(l, b, E_γ) as function of γ-ray energy. First skymap is integrated over full line-of-sight. HIR, H2R signify separated contributions from HI and H_2, each with Galactocentric rings.

Units: emissivity: MeV^2(H atom)$^{-1}$ sr^{-1} s^{-1} MeV^{-1}, skymaps: MeV^2 cm^{-2} sr^{-1} s^{-1} MeV^{-1}

```
ics_isotropic_skymap_50_600203a
ics_skymap_comp_50_600203a
```

Inverse-Compton skymaps(l, b, E_γ) as function of γ-ray energy. First map is sum over all components, second ('comp') has optical, FIR and CMB scattering components separated.

Units: skymaps: MeV^2 cm^{-2} sr^{-1} s^{-1} MeV^{-1}

```
synchrotron_skymap_54_600203a
synchrotron_skymap_Q_54_600203a
synchrotron_skymap_U_54_600203a
synchrotron_skymap_P_54_600203a
```

Synchrotron skymaps(l, b, ν) as function of frequency. Stokes I, Q, U, P. Units: erg cm^{-2} sr^{-1} s^{-1} Hz^{-1}

```
synchrotron_emissivity_54_600203a
synchrotron_emissivity_Q_54_600203a
synchrotron_emissivity_U_54_600203a
```

Synchrotron emissivity (R, z, ν) or (x, y, z, ν). Stokes I, Q, U. Units: erg cm^{-3} sr^{-1} s^{-1} Hz^{-1}

```
synchrotron_skymap_polang_54_600203a
```

Synchrotron polarization angle $(\frac{1}{2}\arctan(U/Q))$ skymaps(l, b, ν). Units: degrees.

```
synchrotron_skymap_polfra_54_600203a
```

Synchrotron polarized fraction (P/I) skymaps(l, b, ν). Units: dimensionless.

`free_free_skymap_54_600203a`

Free-free skymaps(l, b, ν) as function of frequency. Units: erg cm^{-2} sr^{-1} s^{-1} Hz^{-1}

For skymap output format option skymap_format=3 (Healpix) the files have instead names like

`synchrotron_healpix_skymap_54_600203a`

Units: cm^{-2} sr^{-1} s^{-1} MeV^{-1}. **NB unlike (l,b) skymaps, for Healpix the intensity is not multiplied by E_γ^2.**

`ionization_rate_50_600203a`

Ionization rate by nuclei and electrons as a function of position in the Galaxy. Units: s^{-1}

3.5 Running galprop

(This is currently described in the online README file, whose content should be transferred here.)
cd to the directory where is has been built.
Create directory ../FITS and copy the input data (see above) there.
Create directory ../GALDEF and put the galdef file you want to process there e.g.

`galdef_54_600203a`

To run *galprop*, *cd* to the directory where is has been built and type

```
 ./galprop -r 600203a
```

Note that the version number is contained in the executable and *galprop* looks automatically or the corresponding galdef file with the right prefix, here '54'. If the galdef file does not exist, the program issues an error message and stops.

For a first attempt run the provided test case (e.g. test01) which will run fast, and compare the output with that provided in the distribution.

When it has finished, look at the output e.g.

```
ls ../FITS/*_54_600203a
fv ../FITS/pion_decay_skymap_54_600203a
```

or plot it with *galplot*.

To make another run with the same galdef file (maybe with changed parameters) just run again as above, the output will be overwritten (when the run has finished, not before !). To make another run with a new galdef file, run as above with the new galdef ID.

Parallel runs are possible by running in the background, but pipe the screen output to avoid confusion:

```
 ./galprop -r 600203a > log600203a &
 ./galprop -r 600203b > log600203b &
 ./galprop -r 600203c > log600203c &
```

Of course the memory limits must be respected in parallel runs.

3.6 Screen output

The progress of *galprop* is logged on the standard output, which can conveniently be redirected to a log file. The output is mostly self-explanatory. The parameter verbose=0 gives the minimium output, increasing to 1,2 etc. will give much more. In addition, verbose <0 gives debug output for many routines, which is selectable by the verbose value. To find the available debugs, type

grep verbose *.cc

and a list of the available values and their routines will appear, e.g.

```
propagate_particles.cc: if(galdef.verbose==10) particle.secondary_source_function.print();
gen_pi0_decay_emiss.cc: if(galdef.verbose==-478)
  cout<<"  cs_p_HI, cs_p_He, cs_He_HI, cs_He_He ="<<cs_p_HI<<"  "<<cs_p_He<<"  "<<cs_He_HI<<"  "<<c
```

3.7 Test cases

A number of test cases are provided, for each of which the galdef file and the reference output data are available. Also plots to illustrate the results are provided. The test cases can be used to verify the correct running of the code on any platform; they can also be used as a starting point for the cases the user wants to run, by adjusting a few parameters at a time. The propagation parameters of these test cases are based on the optimized model of Strong et al. (2004b) with the CR source distribution and gas distribution of Strong et al. (2004c).

galdef_50_test01 : basic test of all functions: p,He,Be,B,C, electrons, positrons,antiprotons γ-rays. 2D
galdef_50_test02 : all nuclei $Z = 1 - 26$, no γ-rays. 2D
galdef_50_test03 : p,He, electrons, positrons, γ-rays. 2D

4 galdef file parameter explanations

Some parameters default if not specified, but then a warning is issued and the user should provide explicit values. Some parameters are obsolete but left in this description for information when upgrading the galprop version. All parameters are written to the standard output at the start of the program.

The format of a galdef parameter line is

```
<parameter name>     = <parameter value>        <anything else for information, documentation etc>
```

The parameter value starts at or after column 23 after the '='. Anything after the parameter is ignored and is useful to comment on the value. Strings as parameters (e.g. file names) must not contain blanks. Vectors of values must be comma separated without blanks (e.g. 1,2,3,4,5). Values can be ignored (e.g. to keep for reference) just by changing the parameter name

New in trunk. The parameter = value is always split on the = sign and comments can be added manually with the # character. Vectors can be separated with any combination of ',' and ' ' (e.g. 1,2, 3 4,5). Depending on context, anything after the parameter value may or may not be ignored. In number context (integer, floats) it is ignored, but in string and vector context it isn't.

```
xparameter_1         = 1.2              value kept for information
parameter_1          = 1.3              value I want to use here
```

Generally any text can be entered on a line which will be ignored if it does on follow the format. This is useful to document a run.

General

```
Title                = Zh= 4kpc cs=W i=rigidity^-2.43 D=6.10 0.33 dvdz=0 Va=30// pbar test
```

Descriptive title used to identify the run. With commands like "grep Title galdef*" you can get a summary of all runs. Otherwise ignored by the program.

Grid options for Galaxy and spectra

```
 n_spatial_dimensions = 2
```

Specifies whether 2 or 3 spatial dimensions. 2D is cylindrically symmetric (R, z), 3D is (x, y, z) and may be fully asymmetric or with symmetry in x, y, z as specified by the parameter use_symmetry.

```
r_min                =00.0    min r
```

Minimum galactocentric radius (R) for 2D case, in kpc. Normally 0. Ignored for 3D.

```
r_max                    =30.00   max r
```

Maximum galactocentric radius (R) for 2D case, in kpc.

```
dr                       = 1.0    delta r
```

Cell size in galactocentric radius (R) for 2D case, in kpc.

```
z_min                    =-04.0   min z
```

Minimum height for 2D and 3D case, in kpc. In 3D case with use_symmetry = 1 it must be 0, since in this case only $z > 0$ is explicitly computed.

```
z_max                    =+04.0   max z
```

Maximum height for 2D and 3D case, in kpc.

```
dz                       =  0.1   delta z
```

Cell size in z for 2D and 3D case, in kpc.

```
x_min                    = 00.0   min x
```

Minimum x for 3D case, in kpc. In 3D case with use_symmetry = 1 it must be 0, since in this case only $x > 0$ is explicitly computed. Ignored for 2D.

```
x_max                    =+20.0   max x
```

Maximum x for 3D case, in kpc. Ignored for 2D.

```
dx                       =  0.2   delta x
```

Cell size in x for 3D case, in kpc. Ignored for 2D.

```
y_min                    = 00.0   min y
```

See `x_min`, but now for y-axis.

```
y_max                    =+20.0   max y
```

See `x_max`, but now for y-axis.

```
dy                       =  0.2   delta y
```

See `dx`, but now for y-axis.

```
p_min                    =1000    min momentum (MV)
```

Minimum particle momentum in megavolts (MV), case `p_Ekin_grid = p`. NB do not use except for testing.

```
p_max                    =4000    max momentum
```

Maximum particle momentum in megavolts (MV), case `p_Ekin_grid = p`. NB do not use except for testing.

```
p_factor                 =1.20        momentum factor
```

The ratio between successive momentum grid points, case `p_Ekin_grid = p`. The momentum grid is on a logarithmic scale, so that $p[i] = p_{\min} \times p^i_{factor}; p[0] = p_{\min}$. The number of grid points is chosen to give a maximum value close to `p_max`. NB do not use except for testing.

```
Ekin_min                 =1.0e1  min kinetic energy per nucleon (MeV)
```

Minimum particle kinetic energy per nucleon in MeV, case `p_Ekin_grid = Ekin`.

```
Ekin_max              =1.0e7  max kinetic energy per nucleon
```

Maximum particle kinetic energy per nucleon in MeV, case `p_Ekin_grid = Ekin`.

```
Ekin_factor           =1.20       kinetic energy per nucleon factor
```

The ratio between successive kinetic energy per nucleon grid points, case `p_Ekin_grid = Ekin`. The momentum grid is on a logarithmic scale, so that $E_{kin}[i] = E_{kin\ \min} \times E^i_{kin\,factor}; E_{kin}[0] = E_{kin\ \min}$. The number of grid points is chosen to give a maximum value close to `Ekin_max`.

```
p_Ekin_grid           = Ekin         p||Ekin alignment
```

The grid points are arranged so all particles are aligned the same `Ekin` scale; this is very convenient when computing secondary/primary ratios which are always presented in this form. For details see the explanation elsewhere in this document. [The actually propagation calculation is done in terms of momentum since this is physically more natural, but the user does not have to worry about this.] NB only the case `p_Ekin_grid` is useful for nuclei, p-alignment may however be interesting in some cases, hence this option is kept.

```
E_gamma_min           = 0.1     min gamma-ray energy (MeV)
```

Minimum gamma-ray energy (MeV) for diffuse gamma-ray maps.

```
E_gamma_max           = 1.e6    max gamma-ray energy (MeV)
```

Maximum gamma-ray energy (MeV) for diffuse gamma-ray maps.

```
E_gamma_factor        = 10.     gamma-ray energy factor
```

The ratio between successive gamma-ray energy grid points. The energy grid is on a logarithmic scale, so that $E_\gamma[i] = E_{\gamma\ \min} \times E^i_{\gamma\,factor}; E_\gamma[0] = E_{\gamma\ \min}$. The number of grid points is chosen to give a maximum value close to `E_gamma_max`.

```
nu_synch_min          = 1.0e6   min synchrotron frequency (Hz)
```

Minimum frequency (Hz) for synchrotron maps.

```
nu_synch_max          = 1.0e10  max synchrotron frequency (Hz)
```

Maximum frequency (Hz) for synchrotron maps.

```
nu_synch_factor       = 2.0          synchrotron frequency factor
```

The ratio between successive synchrotron frequency grid points. The frequency grid is on a logarithmic scale, so that $\nu_{synch}[i] = \nu_{synch\ \min} \times \nu^i_{synch\,factor}; \nu_{synch}[0] = \nu_{synch\ \min}$. The number of grid points is chosen to give a maximum value close to `nu_synch_max`.

```
long_min              =  0.5    gamma-ray intensity skymap longitude minimum (deg)
```

Minimum longitude for gamma-ray intensity skymaps (degrees). NB Bin centres as in FITS convention.

```
long_max              =359.5    gamma-ray intensity skymap longitude maximum (deg)
```

Maximum longitude for gamma-ray intensity skymaps (degrees). NB Bin centres as in FITS convention.

```
lat_min               =-89.5    gamma-ray intensity skymap latitude  minimum (deg)
```

Minimum latitude for gamma-ray intensity skymaps (degrees). NB Bin centres as in FITS convention.

```
lat_max              =+89.5    gamma-ray intensity skymap latitude  maximum (deg)
```

Maximum latitude for gamma-ray intensity skymaps (degrees). NB Bin centres as in FITS convention.

```
d_long               = 10.     gamma-ray intensity skymap longitude binsize (deg)
```

Binsize in longitude for gamma-ray intensity skymaps (degrees).

```
d_lat                = 10.     gamma-ray intensity skymap latitude  binsize (deg)
```

Binsize in latitude for gamma-ray intensity skymaps (degrees).

```
healpix_order        = 7      order for healpix skymaps.  7 gives ~0.5 deg and it changes by an ord
```

For gamma-ray and synchrotron skymaps. *New in v54.*

```
lat_substep_number   = 1      latitude bin splitting (0,1=no split, 2=split in 2...)
```

Controls gamma-ray skymap computation accuracy. *New in v54.*

```
LoS_step             = 0.01   kpc, Line of Sight (LoS) integration step
```

Controls gamma-ray skymap computation accuracy. from *r586: Controls also synchrotron skymaps. New in v54.*

```
LoS_substep_number   = 1      number of substeps per LoS integration step (0,1=no  substeps)
```

Controls gamma-ray skymap computation accuracy. *New in v54.*

```
los_integarion_mode  = 1      Selects los integration, 1 gives new method, anything else gives old
```

Controls the selected los integration mode. Useful for debugging purposes, new method should always produce more accurate results. *New in trunk*

```
cameraLocation       = 8.5,0,0 #Location of camera for skymap integration (x,y,z)
```

Specifies the location of the camera for skymap integration. Only works properly for `los_integration_mode = 1`. *New in trunk*

Cosmic-ray propagation parameters

```
D0_xx                =6.10e28  diffusion coefficient at reference rigidity
```

The spatial diffusion coefficient divided by $\beta(=v/c)$ at rigidity `D_rigid_br`. The value at other rigidities is determined via the formula $D = \beta D_{0xx}(\rho/D_{rigid\,br})^{D_{g1}}$ for rigidity $<$ `D_rigid_br`, $D = \beta D_{0xx}(\rho/D_{rigid\,br})^{D_{g2}}$ for rigidity $>$ `D_rigid_br`.

```
D_rigid_br           =4.0e3    reference rigidity for diffusion coefficient in MV
```

Rigidity for `D0_xx` formula, and also break point in case `D_g_1 != D_g_2`.

```
D_g_1                = 0.33    diffusion coefficient index below reference rigidity
```

see formula for `D0_xx`. Kolmogorov turbulence corresponds to a value 1/3.

```
D_g_2                = 0.33    diffusion coefficient index above reference rigidity
```

see formula for `D0_xx`.

```
diff_reacc          =1        1=include diffusive reacceleration
```

Flag to indicate whether diffuse reacceleration is to be included in propagation (0=no, $\geq$1=yes). Also controls inclusion of wave-damping: 11=Kolmogorov turbulence, 12=Kraichnan turbulence (see damping parameters below).

```
v_Alfven            =30.      Alfven speed in km s-1
```

Alfvén speed for computation of diffusive reacceleration momentum diffusion coefficient (see explanation elsewhere in document). This parameter is in fact Alfvén speed/$\sqrt{w}$ where w is the ratio of MHD wave energy density to magnetic field energy density, see Strong & Moskalenko (1998).

```
damping_p0          = 1.e6    MV -some rigidity (where CR density is low)
damping_const_G     = 0.02    a const derived from fitting B/C
damping_max_path_L  = 3.e21   Lmax~1 kpc, max free path
```

These parameters refer to the self-consistent MHD wave damping model described in Ptuskin et al. (2006), ApJ.642, astro-ph/0510335. Use of these parameters and the choice of turbulence spectrum is controlled by parameter diff_reacc.

```
convection          =0        1,2=include convection
```

Flag to indicate whether convection is to be included in propagation, and to specify the model for convection.
1: original formulation : $v = v0_{conv} + dv/dz_{conv} * z$
2: *new at r1799.*
$v = +(v0_{conv} + dv/dz_{conv} * (|z| - z0_{conv}))$, $z > +z0_{conv}$,
$v = -(v0_{conv} + dv/dz_{conv} * (|z| - z0_{conv}))$, $z < -z0_{conv}$,
$v = 0, |z| < z0_{conv}$.

```
v0_conv             =0.       starting  convection velocity in km s-1
```

convection=1: Convection velocity at $z = 0$ (km s^{-1}). In practice only a value 0 is physical, from symmetry.
convection=2: Convection velocity at $z = z0_{conv}$ (km s^{-1}).

```
dvdz_conv           =7.       dV/dz=grad V  in km s-1 kpc-1
```

Gradient of convection velocity, assumed linear, in km s^{-1} kpc^{-1}.

```
z0_conv             =1.0      start |z| for wind, kpc
```

Used when convection = 2; see description above. *new at r1799.*

```
nuc_rigid_br        =1.0e2    reference rigidity for primary nucleus injection index in MV
```

In the case that the primary nucleus injection spectra have a break, this defines the rigidity of the break in MV. It it the same for all primary nuclei.

```
nuc_g_1             =2.43     nucleus injection index below reference rigidity
```

Injection index below `nuc_rigid_br`.

```
nuc_g_2             =2.43     nucleus injection index index above reference rigidity
```

Injection index above `nuc_rigid_br`.

```
inj_spectrum_type   =rigidity    rigidity||beta_rig||Etot||dirac nucleon injection spectrum type
```

The primary nucleus injection spectrum may be defined to be a power law in rigidty, $\beta\times$rigidity or total energy. Normally rigidity is used since it corresponds to a power-law in momentum favoured by SNR shock acceleration models. *r603: new type: dirac* A delta-function injection spectrum at energy controlled by nuc_rigidity_break (KE/nucleon) for nuclei and electron_rigid_br (KE) for electrons. Use together with electron_norm_type = 2 or 3 to normalize to injection luminosity. Ditto for proton_norm_type = 2 or 3 but this is normalization is not yet implemented.

```
electron_rigid_br    =1.0e3    reference rigidity for electron injection index in MV
```

In the case that the primary electron injection spectrum has a break, this defines the rigidity of the break in MV.

```
electron_g_1         =2.50     electron injection index below reference rigidity
```

Injection index below `electron_rigid_br`.

```
electron_g_2         =2.50     electron injection index index above reference rigidity
```

Injection index above `electron_rigid_br`.

Parameters controlling interstellar medium.

```
He_H_ratio           =0.11     He/H of ISM, by number
```

Interstellar gas ratio of Helium to Hydrogen. Used for fragmentation, secondary production, energy loss, gamma-ray production. Values 0.08–0.11 are possible, see discussion in Strong & Moskalenko (1998).

```
n_X_CO               = 10       an option to select functional dependence of X_CO=X_CO(R)
  0=constant as below, 9=standard variation as in A&A 2004 paper 10=an exponential
```

Controls choice of X_{CO} variation. *new in v54.*

```
X_CO                 =0.4E20,0.4E20,0.6E20,0.8E20,1.5E20,10.0E20,10.0E20,10.0E20,10.0E20  conversio
       for CO rings  0.0 - 1.5 - 3.5  -  5.5 -  7.5 -  9.5  -  11.5 -  13.5 -  15.5  -  50 kpc
```

Radial dependence of $X_{co} = N(H_2)/W_{co}$. See Strong et al. (2004c) for details. There are 9 values corresponing to the Galactocentric rings indicated. In the case of 9-ring HI,CO data they are all used. In the case of 8-ring data the second and subsequent values are used. *New in v50.*

```
nHI_model            = 1 selection of HI  gas density model  (not yet implemented)
nH2_model            = 1 selection of H2  gas density model  (not yet implemented)

nHII_model           = 3 selection of HII gas density model 1=Cordes et al  1991 2=NE2001
                                                                      3 = Gaensler etal
```

Ionized hydrogen contributes to CR secondary production, energy losses and gamma rays. NE2001 is implemented in a simplified fashion as described in 5.3.1. Option 3 is NE2001 with increased WIM scaleheight and appropriate density according to Gaensler et al 2008, see 5.3.1 (not yet).

```
HII_Te               = 7000  free electron temperature (K)    for free-free absorption
HII_clumping_factor  =  60   free electron clumping factor for free-free absorption
```

Ionized hydrogen (WIM: warm interstellar medium) parameters. Used for radio absorption (synchrotron and free-free) and for free-free radio emission. HII_Te is the electron temperature (K) used in formulae for absorption and emission (default = 7000). HII_clumping_factor is the factor to multipy the absorption coefficient and free-free emissivity relative to the case of a smooth medium. Related to 'filling factor': in the simplest case of equal clouds, HII_clumping_factor = 1/filling-factor. Default = 1. Controlled by parameter free_free_absorption. *from r951*

```
COR_filename         = rbands_co10mm_2001_qdeg.fits.gz
```

File for CO surveys. *new in v54.*

```
HIR_filename         = rbands_hi12_qdeg_zmax1.fits.gz
```

File for HI surveys. *new in v54.*

```
ISRF_file            = ISRF_RMax20_ZMax5_DR0.5_DZ0.1_MW_BB_24092007.fits  (new) input ISRF file
ISRF_filetype        = 1
```

Input file for ISRF, and format type. *new in v54.*

```
ISRF_healpixOrder    = 3 for output of ISRF skymaps
```

Internal size of ISRF healpix skymaps. Used for anisotropic IC calculations. Values greater than or equal to 2 recommended for better accuracy. *new in v54.*

```
ISRF_factors         = 1.0,1.0,1.0        ISRF factors for IC calculation: optical, FIR, CMB
```

Scaling factors for inverse Compton from separate omponents: optical, FIR and CMB. Normally should be 1.0, but other values can be use to experiment. *New in v50.*

```
B_field_name         = halo_dipole   3D models:han ....

Applies if  synchrotron = 2 or 3.
A set of B-field models is implemented. The meanings of the parameters depend on the model.

 B_field_name =     galprop_original : exponential model as in original 'bbbrrrzzz',
                    but using B_field_parameters: Bo (Gauss), rscale (kpc), zscale (kpc).
```

From r592: this model is also used for synchrotron energy losses if synchrotron = 2 or 3, which was not the case before

```
n_B_field_parameters = 10       number of B-field parameters  OBSOLETE

B_field_parameters   = 5.0e-6,10.0,2.0,0,0.0,0.0,0.0,0.0,0.0,0.0,0.0     parameters for 3D models, 
```

Parameters of models specified by name. For the list of models and details of their parameters, see B_field_3D_model.cc The number of parameters is deduced from the input string; current models need up to 10, but more will be required in future. (r885: no longer restricted to exactly 10 parameters). The example above is for exponential model galprop_original, Bo=$5\mu G$, rscale = 10 kpc, zscale = 2 kpc. The models will be documented in this Explanatory Supplement in the future; for now look at the code, it is reasonably self-explanatory.

```
B_field_model        = 050100020   bbbrrrzzz    bbb=10*B(0)  rrr=10*rscale zzz=10*zscale OBSOLETE
```

Only for synchrotron = 0, 1 : Specifies the random magnetic field according to a the law $B = (bbb/10) \times e^{-(R-R_o)/(rrr/10)-z/(zzz/10)}$ microgauss, where $R_o = 8.5$ kpc[5]. e.g., this example has $B = 5e^{(-(R-R_o)/10kpc-z/2kpc)}$ microgauss. *This parameter is deprecated and can be omitted, but description is kept to decipher earlier GALDEF files. Note that synchrotron = 1 uses old routines for synchrotron, which are deprecated. This option is slated for removal in a forthcoming revision. For these reasons please only use instead synchrotron = 2 or 3 and the specification above, with B_field_name = galprop_original.*

Note that synchrotron = 0 uses *this* B-field for synchrotron energy losses even though no synchrotron skymap is requested ! Hence use synchrotron = 2 or 3 always even if a skymap is not desired. In future the B-field specification should be independent of the request for a skymap.

[5] R_o was omitted from the formula in this document although always present in the code. Corrected on 7 Feb 2012

Parameters controlling propagation calculation.

```
fragmentation        =1        1=include fragmentation
```

Flag to indicate whether nuclei fragmentation to be included. Useful for testing effect of fragmentation.

```
momentum_losses      =1        1=include momentum losses
```

Flag to indicate whether momentum losses to be included. Useful for testing effect of losses.

```
radioactive_decay    =1        1=include radioactive decay
```

Flag to indicate whether radioactive decay to be included. Useful for testing effect of decay.

```
start_timestep       =1.0e7   (years)
```

The solution of the propagation equation proceeds starting with a large timestep, repeated typically 20 times (defined by `timestep_repeat`) and reduces the timestep successively. The theory of this accelerated solution is given in Strong & Moskalenko (1998). This parameter is the starting timestep, typically 10^7 years corresponding to the longest timescales in cosmic-ray propagation (20×10^7 years, sufficient for 4 kpc halo).

```
end_timestep         =1.0e1   (years)
```

Final timestep, corresponding to the shortest timescales, dominated by energy-losses; for the highest energy electrons (& positrons) this should be < 100 years for safety, for nuclei-only runs it can be 10^4 years.

```
timestep_factor      =0.50
```

Factor by which timestep is reduced after `timestep_repeat` iterations.

```
timestep_repeat      =20      number of repeats per timestep in  timetep_mode=1
```

Number of iterations of the timestep for each `timestep_factor`. Typically 20. Criterion is that a stable solution is obtained before proceeding to the next smaller timestep, which can be checked by turning on `control_diagnostics` (not normally necessary). NB must be integer format: 10000 not 1e4.

Important note: It is highly recommended to make a test run with start_timestep = end_timestep = a small value like 10^4 *years for nuclei, 100 years for electrons/positrons, and a correspondingly large value for timestep_repeat, like* $10^5, 10^7$*, and timestep_factor=0.9 (the latter is just to terminate the run appropriately), to get the properly converged solution for a total time of* 10^9 *years. This is because the accelerated solution is not always reliable while the small timestep solution is. If the results agree, then parameter scans can be confidently done in the rapid scheme. See Section 'Tests of GALPROP'.*

```
timestep_repeat2     =10000      number of timesteps in timestep\_mode=2
```

At the end of the series of reduced timesteps the propagation can be continued with the smallest timestep. This parameter specifies how many timesteps are to be performed in this mode. This is mainly useful in the 3D mode with stochastic SNR sources, where the large timestep technique is not appropriate since we have a time-dependent problem. If required, the rapid method can be used to get the 'background' solution, while the reduced timesteps follow the subsequent time-dependence at high time resolution. It is also useful to obtain accurate solutions especially with *solution_method* = 2 (see below). This option was only implemented in the 3D case; in 2D this parameter has no effect. r722: now implemented for 2D also.

NB must be integer format: 10000 not 1e4.

```
timestep_print       =200    number of timesteps between printings
```

The full cosmic-ray density array can be printed at intervals using this parameter. NB must be integer format: 10000 not 1e4.

```
timestep_diagnostics =10000  number of timesteps between diagnostics
```

Diagnostics to evaluate the quality of the propagation solution can be generated, normally only occasionally as specified by the number of timesteps between diagnostics. NB must be integer format: 10000 not 1e4. For sample output see Section 'Tests of GALPROP'. Since diagnostics use significant CPU, this parameter should be large for productions runs (100-10000 or more) but can be set to 1 for testing and evaluating the progress of a run in detail.

```
control_diagnostics  =0      control detail of diagnostics
```

Controls the amount of detail in diagnostics. Larger → more detail.

```
solution_convergence = 1     use convergence diagnostic
```

Uses the convergence state return by propel_diagnostics to end a timestep loop and pass to the next smaller timestep (timestep mode 1) or end the propagation loop (timestep mode 2). 1 = do this, 0= don't do this. The criterion is 'no change (to machine accuracy) in the CR flux since the last call of propel_diagnostics', so that further iterations have no effect on the solution with this timestep. Most useful in timestep mode 2, where the number of steps can be set very large and the convergence check automatically ends the propagation at the appropriate number of steps, ensuring an accurate solution without having to experiment manually with the diagnostics.

```
solution_method      = 2     1=method 1:Crank-Nicolson, 2,21 = method 2:fully explicit in time
```

Controls the method to solve the propagation equation, in timestep mode 2 only. New at r705. Standard method is 1=Crank-Nicolson which is also the default if this parameter is absent (upwards compatible). Method 2 is fully explicit in time, see description of routine *propel.cc*. It gives more accurate solutions, which tend to exact according to the *alpha*-based diagnostics, but are not unconditionally stable (while Crank-Nicolson is). For this reason only use for short timesteps (typically 100-1000 years). Since no solution of matrix equations is required, method 2 is faster than 1 for the same timesteps, and this compensates for the need for smaller steps. Specifying 21 invokes a 'turbo' version of method 2, which uses linear arrays instead of 2D or 3D arrays in the propagation code. This leads to significant speed-up even in non-parallel compilations, and is much better adapted to parallelization e.g. via OpenMP. It is available both in 2D and 3D (from r805).

For timestep mode 1, where typically large timesteps are uses at the start, the method is fixed at Crank-Nicolson to ensure stability.

```
network_iterations   =1      number of iterations of entire network     (default 1 if absent)
network_iter_compl   =1      number of iterations for nuclei, including protons (default 1 if absen
network_iter_sec     =1      number of iterations for the entire network (default 1 if absent)
```

The nuclei fragmentation network starts with the largest A and works downwards in A. More most nuclei this ensures that all secondary, tertiary, etc. products are produced in one complete run. So normally `network_iterations=1`. However there may be cases where more iterations are required. An example of those is the β^- decay ^{10}Be → ^{10}B that requires at least 2 network iterations for nuclei. Another example is the calculation of self-consistent damping that requires many iterations using protons only. Generally, secondary leptons require only 1 iterations, hence having `network_iter_sec=1` should be sufficient. They are expensive to calculate and turning them off can greatly increase the speed of calculations. Note that if `network_iterations < network_iter_compl`, `network_iterations` is increased to be equal to `network_iter_compl`.

```
prop_r               = 1  1=propagate in r (2D)
```

Flag to indicate whether to propagate in radial direction (2D only). This and `x,y,z,p` flags are useful for testing the code, e.g., by allowing propagation in one direction only.

```
prop_x               = 1  1=propagate in x (2D)
```

Flag to indicate whether to propagate in x-direction (3D only).

```
prop_y               = 1  1=propagate in y (3D)
```

Flag to indicate whether to propagate in y-direction (3D only).

```
prop_z               = 1  1=propagate in z (2D, 3D)
```

Flag to indicate whether to propagate in z-direction (2D,3D).

```
prop_p               = 1  1=propagate in momentum
```

Flag to indicate whether to propagate in momentum (2D,3D).

```
use_symmetry         = 0  0=no symmetry, 1=optimized symmetry, 2=xyz symmetry by copying(3D)
```

This is only relevant for 3D. The code will solve the 3D case in general, but this leads to large computer resource requirements, in particular large memory. Often symmetry can be assumed without reducing the usefulness of the results since even with stochastic sources the fact that the sources are symmetrically distributed has no effect on the conclusions about, for example, fluctuations. Hence `use_symmetry=1` is generally recommended and the code has been specially optimized to take advantage of this. The memory requirements are reduced by a factor 8 relative to the general case. *Note added 22 Oct 2010, AWS: in the current version use_symmetry = 1 does not work correctly, so use_symmetry = 0 is recommended. Now that computer resources have grown since GALPROP began, the symmetry options are less relevant.*

```
vectorized           = 0  0=unvectorized code, 1=vectorized code
```

Flag to indicate use of vectorized portions of code. Only advantageous on vector processor machines, in 3D. Only applies to `timestep_mode=2`. **Obsolete now that OpenMP has been implemented for mulitple cores**.

```
source_specification = 0  2D::1:r,z=0 2:z=0  3D::1:x,y,z=0 2:z=0 3:x=0 4:y=0
```

This parameter is for testing only, leave at 0.

```
source_model         = 1  CR source model for nuclei 0=zero 1=parameterized 2=Case&B 3=pulsars 4= 5=S&Mattox 6
```

Various CR source distributions. `source_model=1` uses the following 3 source parameters, in the formula $R^{\alpha}e^{-\beta R}$, with cutoff at $R = r_{\max}$ (for details see Strong & Moskalenko, 1998).

```
source_parameters_1  = 0.5    model 1:alpha

source_parameters_2  = 1.0    model 1:beta

source_parameters_3  = 20.0   model 1:rmax

source_parameters_4  =  0.0   model 1:N/A

source_parameters_5  =  0.0   model 1:N/A

source_model_elec    = 1   CR source model for electrons, definitions as for nuclei

source_pars_elec_1 etc : parameters of CR source model for electrons
```

If source_model_elec is absent it will default to source_model. If source_pars_elec_1 etc are absent they will default to source_parameters_1 etc.

```
n_cr_sources         = 0     number of pointlike cosmic-ray sources   3D only!
```

This allows individual CR sources to be inserted in 3D. Useful for testing the response to a delta function, otherwise has not been used much.

```
cr_source_x_01       = 10.0  x position of cosmic-ray source 1 (kpc)
cr_source_y_01       = 10.0  y position of cosmic-ray source 1
cr_source_z_01       = 0.1   z position of cosmic-ray source 1
cr_source_w_01       = 0.1 sigma width  of cosmic-ray source 1
cr_source_L_01       = 1.0   luminosity of cosmic-ray source 1
cr_source_x_02       = 3.0   x position of cosmic-ray source 2
cr_source_y_02       = 4.0   y position of cosmic-ray source 2
cr_source_z_02       = 0.2   z position of cosmic-ray source 2
cr_source_w_02       = 2.4 sigma width  of cosmic-ray source 2
cr_source_L_02       = 2.0   luminosity of cosmic-ray source 2
```

```
SNR_events           = 0    handle stochastic SNR events
```

Flag to indicate whether to generate CR sources in the form of SNR, random in space and time. Only in 3D and only for `timestep_mode=2`. The spatial distribution of SNR follows `source_specification`, the time dependence is determined by the following 2 parameters.

```
SNR_interval         = 1.0e4 time interval in years between SNR in 1 kpc^-3 volume
```

Determines the rate of SNR, according to method described elsewhere in this document. The total SNR rate corresponding to a given value of `SNR_interval` can be obtained by running galprop and noting the value printed on output. Roughly 1.0 years corresponds to 3 SNR/century.

```
SNR_livetime         = 1.0e4 CR-producing live-time in years of an SNR
```

The time in years for which an SNR remains a "live" source of CR.

```
SNR_electron_sdg     = 0.00      delta electron source index Gaussian sigma
```

Allows for dispersion in the electrons source spectral index. This parameter gives the Gaussian sigma of the index around the values specified in `electron_g_1,2`. A value 0.00 produces no dispersion.

```
SNR_nuc_sdg          = 0.00      delta nucleus  source index Gaussian sigma
```

Allows for dispersion in the nuclei source spectral index. This parameter gives the Gaussian sigma of the index around the value specified in `nuc_g_1,2`. A value 0.00 produces no dispersion.

```
SNR_electron_dgpivot = 5.0e3     delta electron source index pivot rigidity (MeV)
```

In the case of dispersion in the electron source index, this parameter defines the rigidity about which the spectrum pivots.

```
SNR_nuc_dgpivot      = 5.0e3     delta nucleus  source index pivot rigidity (MeV)
```

In the case of dispersion in the nucleon source index, this parameter defines the rigidity about which the spectrum pivots.

```
HI_survey            = 9         HI survey 8=orig 8 rings  9=new 9 rings
CO_survey            = 9         CO survey 8=orig 8 rings  9=new 9 rings
```

Choice of HI, CO surveys. v42.2 had 8 rings, v50 includes also improved surveys with 9 rings. *New in v50. not used in v54, replace by input files*

```
proton_norm_Ekin     = 1.00e+5 proton kinetic energy for normalization (MeV)
```

kinetic energy for normalization of proton flux in MeV.

```
proton_norm_flux     = 5.50e-9 flux of protons at normalization energy (cm^-2 sr^-1 s^-1 MeV^-1)
```

proton flux at $R = R_\odot = 8.5$ kpc, $z = 0$, at `proton_norm_Ekin` in cm^{-2} sr^{-1} s^{-1} MeV^{-1}.
Unless proton_norm_type $\neq 1$, see below.

```
proton_norm_type     = 1   type of primary protons and nuclei normalization
```

new parameter in r603 but not yet used. 1 = normalize at solar position as usual in GALPROP. This is the default if this parameter is absent. 0 = no normalization at all (uses arbitrary units for primary source function), 2 = normalize to total proton injection luminosity in particles s^{-1}, 3 = normalize to total proton injection luminosity in erg s^{-1}. The luminosity is evaluated over the whole Galaxy volume. The value of the injection luminosity is specified via proton_norm_flux; typical values 10^{44} particles s^{-1}, 10^{41} erg s^{-1}.

```
electron_norm_Ekin   = 3.45e4  electron kinetic energy for normalization (MeV)
```

kinetic energy for normalization of electron flux, in MeV.

```
electron_norm_flux   = 4.0e-10 flux of electrons at normalization energy (cm^{-2} sr^{-1} s^{-1} MeV^{-1})
```

electron flux at $R = R_\odot = 8.5$ kpc, $z = 0$, at electron_norm_Ekin, in cm^{-2} sr^{-1} s^{-1} MeV^{-1}.
Unless electron_norm_type $\neq 1$, see below.

```
electron_norm_type     = 1   type of primary electrons normalization
```

new parameter in r603, implemented and used. 1 = normalize at solar position as usual in GALPROP. This is the default if this parameter is absent. 0 = no normalization at all (uses arbitrary units for primary source function), 2 = normalize to total electron injection luminosity in particles s^{-1}, 3 = normalize to total electron injection luminosity in erg s^{-1}. The luminosity is evaluated over the whole Galaxy volume. The value of the injection luminosity is specified via electron_norm_flux; typical values 10^{42} particles s^{-1}, 10^{39} erg s^{-1}.

Normally, the flux of CRs is normalized post-propagation to be equal to proton_norm_flux and electron_norm_flux at the energy proton_norm_Ekin and electron_norm_Ekin for nuclei and electrons, respectively. In special cases it may be desired to have absolute normalization of the CR source function. This can be achieved by setting proton_norm_flux and electron_norm_flux to 0 and using source_norm and electron_source_norm instead for protons and electrons, respectively (otherwise these two parameters are not used). The exact units of these variables depend on the chosen source distribution. The value of these parameters for a given proton_norm_flux and electron_norm_flux are given in the full nuclei output from GALPROP. The abundances of other nuclei scale relative to protons via the source abundances as usual.

```
source_norm          = 1.0e-10  absolute normalization for   proton CR source function
                                (only if   proton_norm_flux = 0)

electron_source_norm = 1.0e-10  absolute normalization for electron CR source function
                                (only if electron_norm_flux = 0)

max_Z                = 28     maximum number of nuclei Z listed
```

Specifies how many parameters `use_Z_..` follow.

```
use_Z_1              = 1
```

Flag to specify that nuclei with $Z = 1$ are to be processed.

```
use_Z_2              = 1
```

Flag to specify that nuclei with $Z = 2$ are to be processed, and so on:

```
use_Z_3              = 1
use_Z_4              = 1
use_Z_5              = 1
use_Z_6              = 1
use_Z_7              = 1
use_Z_8              = 1
use_Z_9              = 1
use_Z_10             = 1
use_Z_11             = 1
use_Z_12             = 1
use_Z_13             = 1
use_Z_14             = 1
use_Z_15             = 1
use_Z_16             = 1
use_Z_17             = 1
use_Z_18             = 1
use_Z_19             = 1
use_Z_20             = 1
use_Z_21             = 1
use_Z_22             = 1
use_Z_23             = 1
use_Z_24             = 1
use_Z_25             = 1
use_Z_26             = 1
use_Z_27             = 1
use_Z_28             = 1
use_Z_29             = 0
use_Z_30             = 0
```

The following parameters give the primary source isotopic abundances. The values are relative, the final CR fluxes are normalized so that the proton flux is as specified by `proton_norm_flux`. Hence the user is free to use abundances relative to any species. The abundances are assumed to be valid at a kinetic energy per nucleon equal to `proton_norm_Ekin`. The example abundances give here have been described in SM2001 (COSPAR 2001).

```
iso_abundance_01_001 = 1.430e6        H  Source ELEM.abund.: Meyer,Drury,Ellison 1998,SSRv 86,179
```

abundance of $Z = 1 A = 1$.

```
iso_abundance_02_004 = 1.350e5        He   was 0.069e6   // Solar system relative isotope abund.:
```

abundance of $Z = 2 A = 4$, and so on:

```
iso_abundance_03_006 =    0.          Li
iso_abundance_04_009 =    0.          Be
iso_abundance_05_010 =    0.          B
iso_abundance_06_012 = 2548.  (2573)  C =3000                                     12- 0.955
iso_abundance_06_013 =   25.                                                      13- 0.045
iso_abundance_07_014 =  175.          N  =137.
iso_abundance_08_016 = 3673.          O
iso_abundance_09_019 =    0.          F
iso_abundance_10_020 =  310.   (403)  Ne =???                                     20- 0.88
iso_abundance_10_022 =   93.             22/20 =0.3 in source (DuVernois et al 1996)    22- 0.12
iso_abundance_11_023 =   21.          Na
iso_abundance_12_024 =  626.          Mg =734 *1.1                                24- 0.78
iso_abundance_12_025 =   80.7                                                     25- 0.10
```

```
iso_abundance_12_026 =  100.5                                                         26- 0.12
iso_abundance_13_027 =   45.         Al
iso_abundance_14_028 =  680.   (760) Si =707       Source ab.: Hesse et a. 1996    28/28- 1.00
iso_abundance_14_029 =   60.                                                       29/28- 0.09
iso_abundance_14_030 =   20.                                                       30/28- 0.03
iso_abundance_15_031 =    8.         P  =4.92
iso_abundance_16_032 =   97.0  (105) S  =92.4      Source ab.: Thayer 1997         32/32- 1.00
iso_abundance_16_033 =    2.1                                                      33/32- 0.026
iso_abundance_16_034 =    6.3                                                      34/32- 0.062
iso_abundance_17_035 =    0.9        Cl
iso_abundance_17_037 =    3.2
iso_abundance_18_036 =   20.0        Ar =15.2
iso_abundance_18_038 =    4.0           -introduced by imos
iso_abundance_19_039 =    0.         K  -introduced by imos
iso_abundance_20_040 =   39.0        Ca =42.
iso_abundance_20_041 =    0.8
iso_abundance_21_045 =    0.         Sc
iso_abundance_22_046 =    0.         Ti -introduced by imos
iso_abundance_22_047 =    1.9
iso_abundance_22_048 =    7.3
iso_abundance_22_049 =    0.
iso_abundance_22_050 =    1.4
iso_abundance_23_051 =    0.         V
iso_abundance_24_050 =    1.4        Cr -introduced by imos
iso_abundance_24_051 =    0.
iso_abundance_24_052 =    7.7
iso_abundance_24_053 =    3.4
iso_abundance_24_054 =    1.0
iso_abundance_25_053 =    2.2        Mn -introduced by imos
iso_abundance_25_055 =    6.8
iso_abundance_26_054 =   72.1  (882) Fe =713  Source abund.: Connell & Simpson 1997 54/56= 9.3%
iso_abundance_26_055 =   12.1                                                      55/56= 1.6%
iso_abundance_26_056 =  776.1                                                      56/56= 100%
iso_abundance_26_057 =   28.8                                                      57/56= 3.7%
iso_abundance_26_058 =    1.41                                                     58/56= 0.18%
iso_abundance_27_059 =    0.4        Co =1.28
iso_abundance_28_058 =   30.61 (44.3) Ni =40.2 Source ab.: Connell & Simpson 1997    58/58= 100%
iso_abundance_28_059 =    0.81                                                     59/58= 2.6% =0
iso_abundance_28_060 =   13.17                                                     60/58=43.2%
iso_abundance_28_061 =    0.35                                                     61/58 <1.2%
iso_abundance_28_062 =    1.62                                                     62.58= 5.4%
```

```
total_cross_section  =   2    =0 -Letaw83; =1 -WA96 Z>5 & BP01 Z<6; =2 -BP01 (2-best)
```

Options for determining total fragmentation cross sections. BP= code from V.S.Barashenkov,A.Polanski and this is the preferred option.

```
cross_section_option =   012    100*i+j  i=1: use Heinbach-Simon C,O->B j=kopt j=11=Webber, 21=ST
```

Options for determining cross sections. Details on kopt from nuc_package.cc:

```
//3 kopt  =0 - uses best alghorithm described in comments below (not recommended);
//3       =1 - forces to use Webber'93 code (no renormalization etc.);
//3       =2 - forces to use TS00 code (no renormalization etc.);
//3       =3 - forces to use a const cross section fitted to the data.
//3       =10- forces to use Webber'93 code (renormalized if data exists);
```

```
//3       =11- forces to use cross section fit if exists (otherwise equiv. 10);
//3       =12- forces to use a numerical table if exists (otherwise equiv. 11);
//3       =20- forces to use TS00 code (renormalized if data exists).
//3       =21- forces to use cross section fit if exists (otherwise equiv. 20).
//3       =22- forces to use a numerical table if exists (otherwise equiv. 21);
//3 The best values recommended are kopt = 12, 22 (12 is preferrable).
//3 uses file_no[1] and file_no[3] as indicators of the data array and fit params.
```

Example: For kopt=22, the code will first try to find the cross section of a given channel in the tables in `eval_iso_cs.dat`. If it fails, it will fall back to kopt=21 and look for a fit in `p_cs_fits.dat`. If no fit is found, kopt=20 will be used, and the cross section will be calculated using the TS00 formula, re-normalized to fit the data in `isotope_cs.dat`.

The file `eval_iso_cs.dat` contains calculated and measured cross sections as data tables suitable for interpolation; `p_cs_fits.dat` contains parameters of fits to measured cross sections and `isotope_cs.dat` contains individual experimental data points. References to data sources are included in the comments within these data files.

```
primary_electrons    = 0
```

Flag to indicate whether to propagate primary electrons.

```
secondary_positrons  = 0
```

Flag to indicate whether to propagate secondary positrons.

```
secondary_electrons  = 0
```

Flag to indicate whether to propagate secondary electrons.

```
secondary_antiproton = 2   1=uses nuclear scaling; 2=uses nuclear factors by Simon et al 1998
```

Flag to indicate whether to propagate secondary antiprotons, with 2 options.

```
tertiary_antiproton  = 1
```

Flag to indicate whether to propagate tertiary antiprotons.

```
secondary_protons    = 1
```

Flag to indicate whether to propagate secondary protons.

```
gamma_rays           = 0    1=compute gamma rays 2=also for HI,CO rings
```

Flag to indicate whether to compute diffuse Galactic gamma-ray skymaps and emissivities Requires computation of protons, Helium and electrons. If option 1 is chosen, only the line-of-sight integrated skymaps are produced, for each component bremsstrahlung, pion-decay and inverse Compton.

With option 2 (*New in v50*) the skymaps of bremsstrahlung and pion-decay for each HI, CO ring are output in addition. This produces 4 extra output datasets: bremss+HI,bremss+H_2,pion+HI,pion+H_2. This is useful for studies of CR distribution and X_{co}.

```
pi0_decay            = 3     1= old formalism 2=Blattnig et al. 3=Kamae et al. 4=Huang etal.
```

Options for formalism used to compute pion-decay gamma rays, secondary positrons and electrons. In this version Kamae (2006) only used for gamma rays, and only for p-p interactions, otherwise old method is still used. *from r1265:* 4 = use Huang etal 2007 (Astroparticle Physics, 27, 429) matrices. These use ISM composition so the ISM He fraction (parameter He_H_ratio) is not used in this case, but the composition is consistent (He/H 10% by number). See text for further details.

```
IC_anisotropic       = 0    1=compute anisotropic IC
```

Flag to indicate whether to compute anisotropic inverse Compton scattering according to MS2000. NB: takes lots of CPU, best in parallel mode.

```
bremss                 = 1    1=compute bremsstrahlung
```

from r1265: 2 = use C++ version instead of fortran. This is a translation from the fortran version, but also with its own energy integration. Gives (almost) identical results. Will be used for future developments of the bremsstrahlung physics treatment.

```
synchrotron            = 2    1,2,3 = compute synchrotron skymaps
```

Flag to indicate whether to compute synchrotron skymaps. Also indicates which B-field is to be used to compute synchrotron energy losses of electrons and positrons, independent of whether synchrotron skymaps are computed. 0= old B-field format (deprecated) for synchrotron losses, but no synchrotron skymaps. 1= old B-field format (deprecated), 2 = use B_field_model, B_field_parameters. For more details see those parameters. 3 = as 2 but compute Stokes vectors in addition (under development). Synchrotron skymaps require `primary_electrons =1 and/or secondary_electrons =1 and/or secondary_positrons =1`.

```
free_free_absorption = 1      0=no absorption 1 free-free absorption of synchrotron
                              2=replace synch total with free-free
```

Controls whether to apply free-free absorption of synchrotron and free-free emission. Parameters nHII_Te and nHII_clumping_factor control the formulae used. Default = 0. Setting this parameter to 2 computes free-free emission and outputs the radio skymaps instead of total synchrotron (Healpix only) in synchrotron_healpix_GALDEF_ID. *from r951*

DM options

```
comment                = the dark matter (DM) switches and user-defined parameters
DM_positrons           = 0   1=compute DM positrons
DM_electrons           = 0   1=compute DM electrons
DM_antiprotons         = 0   1=compute DM antiprotons
DM_gammas              = 0   1=compute DM gammas

DM_double0             = 2.8    core radius, kpc
DM_double1             = 0.43   local DM mass density, GeV cm-3
DM_double2             = 80.    neutralino mass, GeV
DM_double3             = 40.    positron width distribution, GeV
DM_double4             = 40.    positron branching
DM_double5             = 40.    electron width distribution, GeV
DM_double6             = 30.    electron branching
DM_double7             = 50.    pbar width distribution, GeV
DM_double8             = 40.    pbar branching
DM_double9             =3.e-25  <cross_sec*V>-thermally overaged, cm3 s-1

DM_int0                = 1    isothermal profile
DM_int1                = 1
DM_int2                = 1
DM_int3                = 1
DM_int4                = 1
DM_int5                = 1
DM_int6                = 1
DM_int7                = 1
DM_int7                = 1
DM_int8                = 1
DM_int9                = 1
```

Output options

```
skymap_format        = 3         fitsfile format: 0=old format (the default), 1=mapcube for Fermi
```

Format for output gamma/synchroton skymaps. 0=(l,b,E) as in previous versions, 1= format compatible with Fermi science tools, 3=Healpix (recommended)

```
output_gcr_full      = 0  output full galactic cosmic ray array
```

Flag to indicate whether to output the full (R, z, p) or (x, y, z, p) array of nuclei. For 3D this array can be very large but is required for studying spatial effects. It must be set to 1 if subsequent runs (`warm_start`) are to be made using this run as input. It is also required if the cosmic-ray spectra are to be plotted using the *galplot* package.

Run time options

```
warm_start           = 0  read in nuclei file and continue run
```

Flag to indicate the run is to be started with results output from previous run with `output_gcr_full =1`. Useful for long runs which have to be broken up due to CPU time limitations. NB the galdef file should be the same as for the first run, apart from this parameter; the only possible differences should be the values of `end_timestep` and `timestep_repeat2`, and the generation of gamma-rays etc. The grid and selected nuclei, electrons etc. must be the same.

```
verbose              = 0  verbosity: 0=min,10=max
```

Controls level of output: 0,1,2...10.

```
test_suite           = 0  run test suite instead of normal run
```

Instead of normal run, generates test output on many subroutines such as cross-section evaluation.

5 Description of Basic Routines

This refers mainly to v50 and needs updating for v54.

5.1 `galdef.read`

Reads the *galdef* file and assigns initial parameter values.

5.2 `read_nucdata & set_sigma_cc`

5.3 `create_galaxy`

Creates spatial grid and defines distributions of gas (H_2, HI, HII — `nH2, nH2_av, nHI, nHI_av, nHII, nHII_av`), magnetic field (`B_field_model`), reads the interstellar radiation field from a file (`read_isrf`) and defines the ISRF energy density (`gen_isrf_energy_density`), creates supernova remnants (`create_SNR`), and defines the skymap parameters for gamma rays and synchrotron emission.

5.3.1 `nH2, nH2_av, nHI, nHI_av, nHII, nHII_av`

The routines `nH2, nHI` define the gas number densities in the form of a table, which gives the gas densities in the Galactic plane. The extention of the gas distribution to an arbitrary height above the plane is made using some analytical approximations. The routine `nHII` uses only analytical approximations.

The routines `nH2_av, nHI_av, nHII_av` provide a calculation of an average gas density over a step in the z-grid using a smaller step.

nH2: The routine calculates H_2 number density in mol cm^{-3}

$$n_{H_2}(R,Z) = \epsilon_0(R) X\, e^{-\ln 2 (Z-z_0)^2/z_h^2} \quad cm^{-2}\ kpc^{-1}, \tag{5}$$

where $\epsilon_0(R)$ (K km s^{-}1) – CO volume emissivity, and $z_0(R)$, $z_h(R)$ – the height scale and width are taken from Bronfman et al. (1988, Table 3/Cols 4,7,10), and $X = n_{H_2}/n_{CO} = 1.9 \times 10^{20}$ is the conversion factor taken from Strong & Mattox (1996).

nHI: The routine calculates H_I number density in atom cm^{-3}. The relative distribution is from Gordon & Burton (1976, Table 1), but renormalized to agree with Dickey & Lockman (1990), where on page 252 they give their best model for the z-distribution and state the total integral perpendicular to the plane $= 6.2 \times 10^{20}$ cm^{-2}:

$$n_{H_I}(R,Z) = Y(R) f(Z), \tag{6}$$

where $Y(R)$ is the renormalized distribution by Gordon & Burton ($R < 16$ kpc), and $f(Z)$ is the z-dependence which is calculated following Dickey & Lockman for $R < 8$ kpc, following Cox et al. (1986) for $R > 10$ kpc, and interpolated in between. For $R > 16$ kpc the exponential tail is assumed with scale high 3 kpc.

nHII: The ionized component currently calculated using a cylindrically symmetrical model based on NE2001 by Cordes et al. (1991, Eq.(6) and Table 1). More sophisticated 3D model for the inner Galaxy is developed by Taylor & Cordes (1993) and outer Galaxy by Lazio & Cordes (1998a). Galactic center region has been considered by Lazio & Cordes (1998b). These latter models can be used in future for 3D calculations. A revised scale-height using more pulsar DMs was determined by Gaensler et al. (2008) which has been implemented as option to replace this component in NE2001. *more details required.*

5.3.2 `read_isrf`

Reads 3D interstellar radiation field (ISRF) from a file, currenty `isrf_interp_04_000015`. The units of the ISRF stored is $[\lambda U_\lambda] = \mu$ eV cm^{-3} μ^{-1}. In the routine `read_isrf` the units are changed to $[\nu U_\nu] =$ Hz eV cm^{-3} Hz^{-1}.

5.3.3 `gen_isrf_energy_density`

5.4 `create_gcr`

5.5 `propagate_particles`

The routine organizes a loop over the cosmic ray species: calls a routine to create transport arrays, generates secondary source, calls `propel` to propagate particles and finally normalizes nuclei and electrons.

5.5.1 `create_transport_arrays`

Generates arrays for use by `propel` for the given species: assigns primary source function, diffusion coefficient, fragmentation rate, momentum loss rate, decay rate. Normalizes source spectra according to abundances, interpreting as flux at the same E_{kin} (*kinetic energy per nucleon*) or at the same p_1 (*momentum per nucleon*).

Normalization of the nucleon primary source function: The source function is assumed to be a power-law in rigidity ρ with reference value at ρ_{br}

$$q_A(p) = c_A \left(\frac{\rho}{\rho_{br}}\right)^{-\gamma}, \tag{7}$$

and the source abundance is defined as the ratio

$$X = \frac{Q_A(p_1)}{Q_1(p_1)} = \frac{A q_A(p_1)}{q_1(p_1)} \tag{8}$$

where q_1 refers to protons.

Since $\rho/\rho_{br} = \frac{1}{Z}(p_A/\rho_{br})$ and $p_A = Ap_1$, one can get

$$\begin{aligned} X &= \frac{Ac_A}{c_1}\left(\frac{p_A}{Z\rho_{br}}\right)^{-\gamma}\left(\frac{p_1}{\rho_{br}}\right)^{\gamma} = \frac{c_A A^{1-\gamma} Z^{\gamma}}{c_1}, \\ \frac{c_A}{c_1} &= X A^{\gamma-1} Z^{-\gamma}. \end{aligned} \tag{9}$$

The factor c_A/c_1 is applied at the end of the source function generation.

The global normalization to the absolute proton flux is applied at the end of the entire propagation, in `nuclei_normalize`. A reference list of abundances relative to protons can be found in Meyer, Drury, & Ellison (1998, Table 1) and is used as the baseline set in the standard *galdef* files.

ρ_{br} can be used as a break rigidity with different exponents above and below, and the formulation ensures continuity at this rigidity. More general method, maybe to be implemented later, valid for any spectrum (above is only for power-law in rid. dity).

Assigning fragmentation rate

The fragmentation rate is assigned using the total cross section calculation on proton target (`in nucleon_cs_cc`), cross section on He target is calculated using phenomenological scaling (`He_to_H_CS`).

In case of protons "fragmentation" means inelastic scattering in which particle losses a substantial part of its energy. In case of antiprotons "framentation" means inelastic scattering plus annihilation. Finally, the fragmentation rate is calculated in every spatial and energy point as:

$$F = \beta c(n_{H_2} + n_{HI} + n_{HII})(\sigma_p + \sigma_{He}\frac{n_{He}}{n_H}), \quad s^{-1}, \tag{10}$$

where $n_H = n_{H_2} + n_{HI} + n_{HII}$.

Energy losses

The energy losses for nucleons, ionization and Coulomb losses, are calculated in `nucleon_loss`. Those for electrons, ionization, Coulomb losses, bremsstrahlung, syncrotron, and Compton losses (Thompson scattering), are calculated in `electron_loss`. The routine `e_KN_loss` allows to calculate Klein-Nishina energy losses.

Klein-Nishina energy losses are calculated using ISRF calculated separately and read by routine `read_isrf`. The ISRF units are $[\nu U_\nu]$ = Hz eV cm^{-3} Hz^{-1}. The energy losses in every spacial point can be calculated as

$$\frac{dp}{dt} = \int d\nu\, \frac{U_\nu}{\hbar\nu}\frac{dp}{dt}(\nu,\epsilon) = \int d(\log\nu)\, \frac{\nu U_\nu}{\hbar\nu}\frac{dp}{dt}(\nu,\gamma), \quad eV\ s^{-1}, \tag{11}$$

where γ is the electron Lorentz-factor, and $dp(\nu,\gamma)/dt$ is calculated in `e_loss_compton`.

5.5.2 `gen_secondary_source`

Combines calculation of all the secondary source functions.

<u>`gen_secondary_positron_source`: secondary e^+ and e^-</u>

The routine `PP_MESON` written in `FORTRAN-77` is designed to calculate the secondary positron (or sec. electron) production spectrum vs. energy (barn/GeV). Positron/electron energy and total nucleus momentum are used as input parameters as well as beam and target nuclei atomic numbers.

The secondary positron/electron source function as used in *galprop* is defined as following (cm^{-3} s^{-1} sr^{-1} MeV^{-1}):

$$q_e(E_{tot}) = \frac{c}{4\pi}\frac{dn(p)}{dt} = \frac{c}{4\pi}\sum_{i=H,He} n_i \sum_j \int dp'\, \beta n_j(p')\frac{d\sigma_{ij}(E_{tot},p')}{dE_{tot}}, \tag{12}$$

where n_i is the gas density, $d\sigma_i j(E_{tot},p')/dE_{tot}$ is the production cross section, $n_j(p')$ is the CR species *density*, and p' is the total momentum of a nucleus. Substitution of dp' with $d(\log E'_{kin})$ gives:

$$q_e(E_{tot}) = \frac{c}{4\pi}A\sum_{i=H,He} n_i \int d(\log E'_{kin})\, E'_{kin}\sum_j n_j(E'_{kin})\frac{d\sigma_{ij}(E_{tot},E'_{kin})}{dE_{tot}}$$

$$= \frac{c}{4\pi} A\,\Delta(\log E'_{kin}) \sum_{i=H,He} n_i \sum_{E'_{kin}} E'_{kin} \sum_j n_j(E'_{kin}) \frac{d\sigma_{ij}(E_{tot}, E'_{kin})}{dE_{tot}}, \tag{13}$$

where we used $dp' = \frac{1}{\beta} A E'_{kin} d(\log E'_{kin})$.

Since positron/electron is assumed massless $E_{tot} = p$. The units should be transferred to cm^2/MeV, a factor applied is 10^{-27}.

`gen_secondary_antiprotons_source`

The routine `ANTIPROTON` written in `FORTRAN-77` is designed to calculate the antiproton (+antineutron) production spectrum vs. momentum (barn/GeV). Antiproton momentum and nucleus momentum per nucleon are used as input parameters as well as beam and target nuclei atomic numbers.

The antiproton source function as used in *galprop* is defined as following (cm^{-3} s^{-1} sr^{-1} MeV^{-1}):

$$q_{\bar{p}}(p) = \frac{c}{4\pi} \frac{dn(p)}{dt} = \frac{c}{4\pi} \sum_{i=H,He} n_i \sum_j \int dp'\, \beta n_j(p') \frac{d\sigma_{ij}(p, p')}{dp}, \tag{14}$$

where n_i is the gas density, $d\sigma_i j(p, p')/dp$ is the production cross section, $n_j(p')$ is the CR species *density*, and p' is the total momentum of a nucleus. Substitution of dp' with $d(\log E'_{kin})$ gives:

$$\begin{aligned} q_{\bar{p}}(p) &= \frac{c}{4\pi} A \sum_{i=H,He} n_i \int d(\log E'_{kin})\, E'_{kin} \sum_j n_j(E'_{kin}) \frac{d\sigma_{ij}(p, E'_{kin})}{dp} \\ &= \frac{c}{4\pi} A\,\Delta(\log E'_{kin}) \sum_{i=H,He} n_i \sum_{E'_{kin}} E'_{kin} \sum_j n_j(E'_{kin}) \frac{d\sigma_{ij}(p, E'_{kin})}{dp}, \end{aligned} \tag{15}$$

where we used $dp' = \frac{1}{\beta} A E'_{kin} d(\log E'_{kin})$.

The units should be transferred to cm^2/MeV, a factor applied is 10^{-27}.

`gen_secondary_protons_source` and `gen_tertiary_antiprotons_source`

`Gen_secondary_protons_source and gen_tertiary_antiprotons_source` are made very similar using the same formalism. The secondary protons and tertiary antiprotons are essentially those which survived after inelastic scattering with protons and He nuclei of the interstellar gas. It is convenient and correct to consider them as additional populations of particles.

The tertiary $\bar{p}$'s are calculated using

$$q_{\mathrm{tert.}\bar{p}}(p) = c\Delta(\log E'_{kin}) \sum_{i=H,He} n_i \sum_{E'_{kin}} E'_{kin} n_{\mathrm{sec.}\bar{p}}(E'_{kin}) \frac{d\sigma_i(p, E'_{kin})}{dp}, \tag{16}$$

where $d\sigma_i(p, E'_{kin})/dp$ is the "production cross section" of tertiary $\bar{p}$'s. It can be further expanded as

$$\frac{d\sigma_i(p, E'_{kin})}{dp} = \sigma_i^{\mathrm{non}}(E'_{kin}) \beta \frac{dN}{dE_{kin}}, \tag{17}$$

where $\sigma_i^{\mathrm{non}}(E'_{kin})$ is the total non-annihilation inelastic cross section, and dN/dE_{kin} is the distribution of $\bar{p}$'s after scattering. Using an approximation given by Tan & Ng (1983):

$$\frac{dN(E'_{kin}, E_{kin})}{dE_{kin}} = \frac{1}{E'_{kin}}. \tag{18}$$

The final expression can be obtained by combining eqs. (16)-(18).

secondary source for nuclei: `decayed_cross_sections`

The formalism is similar to the above, except that the cross section of the disintegration process is convenient to write in energy or momentum per nucleon:

$$\frac{d\sigma_{ij}(p, p')}{dp} = \sigma(p')\, \delta\left(p - \frac{A_{sec}}{A_{prim}} p'\right), \tag{19}$$

where A_{prim}, A_{sec} are the atomic numbers of primary and secondary nucleus, correspondingly. Integrating over dp' then yield a factor A_{prim}/A_{sec}:

$$q_{sec}(p) \quad = \quad \beta c \sum_{i=H,He} n_i \sum_j \frac{A_j}{A_{sec}} \sigma\left(\frac{A_j p}{A_{sec}}\right) n_j \left(\frac{A_j p}{A_{sec}}\right). \tag{20}$$

The `decayed_cross_sections` routine calculates the cross section σ of a given channel taking into account short-lived intermediate states and nuclear reaction network.

5.5.3 `propel`

The basic routine for calculation of the finite-differencing coefficients and numerical solution of the propagation equation.

Numerical solution of propagation equation

Recall the propagation equation:

$$\frac{\partial\psi}{\partial t} = q(\vec{r},p) + \vec{\nabla}\cdot(D_{xx}\vec{\nabla}\psi - \vec{V}\psi) + \frac{\partial}{\partial p}p^2 D_{pp}\frac{\partial}{\partial p}\frac{1}{p^2}\psi - \frac{\partial}{\partial p}\left[\dot{p}\psi - \frac{p}{3}(\vec{\nabla}\cdot\vec{V})\psi\right] - \frac{1}{\tau_f}\psi - \frac{1}{\tau_r}\psi\ , \tag{21}$$

Full explicit method. The diffusion, reacceleration, convection and loss terms in eq. (46) can all be finite-differenced for each dimension (R, z, p) or (x, y, z, p) in the form

$$\frac{\partial\psi_i}{\partial t} = \frac{\psi_i^{t+\Delta t} - \psi_i^t}{\Delta t} = \frac{\alpha_1\psi_{i-1}^t - \alpha_2\psi_i^t + \alpha_3\psi_{i+1}^t}{\Delta t} + q_i\ , \tag{22}$$

where all terms are functions of (R, z, p) or (x, y, z, p).

This is the *fully time-explicit method* (Press et al., 1992) where the updating scheme is

$$\psi_i^{t+\Delta t} = \psi_i^t + \alpha_1\psi_{i-1}^t - \alpha_2\psi_i^t + \alpha_3\psi_{i+1}^t + q_i\Delta t\ . \tag{23}$$

which generalizes simply to any number of dimensions since all the quantities are known from the current step. The *explicit* method has been implemented at r705. It is controlled by the galdef parameter solution_method (see parameter descriptions). It gives more accurate solutions, which tend to exact according to the *alpha*-based diagnostics, but are not unconditionally stable (while Crank-Nicolson is). For this reason it is only applicable for short enough timesteps. Since no solution of matrix equations is required, this method is faster than Crank-Nicolson for the same timesteps, and this compensates for the need for smaller steps.

Fully implicit method. The diffusion, reacceleration, convection and loss terms in eq. (46) can all be finite-differenced for each dimension (R, z, p) or (x, y, z, p) in the form

$$\frac{\partial\psi_i}{\partial t} = \frac{\psi_i^{t+\Delta t} - \psi_i^t}{\Delta t} = \frac{\alpha_1\psi_{i-1}^{t+\Delta t} - \alpha_2\psi_i^{t+\Delta t} + \alpha_3\psi_{i+1}^{t+\Delta t}}{\Delta t} + q_i\ , \tag{24}$$

where all terms are functions of (R, z, p) or (x, y, z, p).

This is the *fully time-implicit method* (Press et al., 1992) where the updating scheme is

$$\psi_i^{t+\Delta t} = \psi_i^t + \alpha_1\psi_{i-1}^{t+\Delta t} - \alpha_2\psi_i^{t+\Delta t} + \alpha_3\psi_{i+1}^{t+\Delta t} + q_i\Delta t\ . \tag{25}$$

This method is unconditionally stable for all α and Δt, but is only 1st-order accurate in time.

The tridiagonal system of equations

$$-\alpha_1\psi_{i-1}^{t+\Delta t} + (1+\alpha_2)\psi_i^{t+\Delta t} - \alpha_3\psi_{i+1}^{t+\Delta t} = \psi_i^t + q_i\Delta t, \tag{26}$$

is solved for the $\psi_i^{t+\Delta t}$ by the standard method (Press et al., 1992). Note that for energy losses we use ‘upwind’ differencing to enhance stability, which is possible since we have only *loss* terms (adiabatic energy *gain* is not included here).

Crank-Nicolson method Alternatively, the propagation equation can be finite-differenced in the form

$$\frac{\partial \psi_i}{\partial t} = \frac{\psi_i^{t+\Delta t} - \psi_i^t}{\Delta t} = \frac{\alpha_1 \psi_{i-1}^{t+\Delta t} - \alpha_2 \psi_i^{t+\Delta t} + \alpha_3 \psi_{i+1}^{t+\Delta t}}{2\Delta t} + \frac{\alpha_1 \psi_{i-1}^{t} - \alpha_2 \psi_i^{t} + \alpha_3 \psi_{i+1}^{t}}{2\Delta t} + q_i \quad (27)$$

This is the *Crank-Nicolson method* (Press et al., 1992) where the updating scheme is

$$\psi_i^{t+\Delta t} = \psi_i^t + \frac{\alpha_1}{2}\psi_{i-1}^{t+\Delta t} - \frac{\alpha_2}{2}\psi_i^{t+\Delta t} + \frac{\alpha_3}{2}\psi_{i+1}^{t+\Delta t} + \frac{\alpha_1}{2}\psi_{i-1}^{t} - \frac{\alpha_2}{2}\psi_i^{t} + \frac{\alpha_3}{2}\psi_{i+1}^{t} + q_i \Delta t \ . \quad (28)$$

It thus uses a combination of implicit and explicit terms, forming the time-average of the differentials. Like the fully implicit method, this method is unconditionally stable for all α and Δt, but is 2nd-order accurate in time, so that larger time-steps are possible than with the 1st-order scheme.

The tridiagonal system of equations

$$-\frac{\alpha_1}{2}\psi_{i-1}^{t+\Delta t} + (1+\frac{\alpha_2}{2})\psi_i^{t+\Delta t} - \frac{\alpha_3}{2}\psi_{i+1}^{t+\Delta t} = \psi_i^t + q_i\Delta t + \frac{\alpha_1}{2}\psi_{i-1}^{t} - \frac{\alpha_2}{2}\psi_i^{t} + \frac{\alpha_3}{2}\psi_{i+1}^{t} \quad (29)$$

or

$$-\frac{\alpha_1}{2}\psi_{i-1}^{t+\Delta t} + (1+\frac{\alpha_2}{2})\psi_i^{t+\Delta t} - \frac{\alpha_3}{2}\psi_{i+1}^{t+\Delta t} = \frac{\alpha_1}{2}\psi_{i-1}^{t} + (1-\frac{\alpha_2}{2})\psi_i^{t} + \frac{\alpha_3}{2}\psi_{i+1}^{t} + q_i\Delta t \quad (30)$$

is again solved for the $\psi_i^{t+\Delta t}$ by the standard method. Note that the RHS has all known quantities from the current time-step.

GALPROP uses the Crank-Nicolson method. Our original paper (Strong & Moskalenko, 1998) describes the fully implicit method, which was in fact never used.

Multidimensional case. The Crank-Nicolson method described above applies to a one-dimensional case; the application to 2 or 3 spatial and one momentum dimension requires a generalization. A straightforward expansion to more dimensions implies solving large matrix equations (no longer tridiagonal); instead the so-called ADI (alternating direction implicit) method is used, in which the implicit solution is applied to each dimension in turn. Each application uses just the operator for that dimension, so the tridiagonal scheme can still be used. This however is not completely valid since it solves a slightly different problem from that with the full operator; however for small enough timesteps the solution is accurate (see Section on Tests of GALPROP). Other possible strategies are given in Press (1992), which could be considered in future.

The *explicit* method, where the full operator can be used in each timestep without any overhead for solving matrix equations, is also useful for obtaining an accurate solution at the end of a run. Although it is not unconditionally stable, this does not matter provided the timesteps are small enough, which is in any case required for the implicit methods to maximise their accuracy. A suitable mix of explicit and implicit methods to obtain an accurate solution with minimum computing requirements, is the goal. The *explicit* method has been implemented at r705. It is controlled by the galdef parameter solution_method (see parameter descriptions). It gives demonstrably more accurate solutions, which tend to exact according to the *alpha*-based diagnostics, but are not unconditionally stable (while Crank-Nicolson is). For this reason only use for short timesteps (typically 100-1000 years). Since no solution of matrix equations is required, method 2 is faster than 1 for the same timesteps, and this compensates for the need for smaller steps.

Boundary conditions. For 2D, three spatial boundary conditions

$$\psi(R, z_h, p) = \psi(R, -z_h, p) = \psi(R_h, z, p) = 0 \quad (31)$$

are imposed at each iteration. No boundary conditions are imposed or required at $R = 0$ or in p. Grid intervals are typically $\Delta R = 1$ kpc, $\Delta z = 0.1$ kpc; for p a logarithmic scale with ratio typically 1.2 is used. Although the model is symmetric around $z = 0$ the solution is generated for $-z_h < z < z_h$ since this is required for the tridiagonal system to be valid.

For 3D, the spatial boundary conditions

$$\psi(\pm x_h, y, z, p) = \psi(x, \pm y_h, z, p) = \psi(x, y, \pm z_h, p) = 0 \quad (32)$$

are imposed at each iteration. Again no boundary conditions are imposed in p. Grid intervals are typically $\Delta x = \Delta y = 0.5$ kpc, $\Delta z = 0.1$ kpc.

Table 1: Coefficients for the finite-differencing scheme in 2 spatial dimensions and one momentum dimension.

Process	Coordinate	$\alpha_1/\Delta t$	$\alpha_2/\Delta t$	$\alpha_3/\Delta t$
Diffusion	R	$D_{xx}\frac{2R_i-\Delta R}{2R_i(\Delta R)^2}$	$D_{xx}\frac{2R_i}{R_i(\Delta R)^2}$	$D_{xx}\frac{2R_i+\Delta R}{2R_i(\Delta R)^2}$
	z	$D_{xx}/(\Delta z)^2$	$2D_{xx}/(\Delta z)^2$	$D_{xx}/(\Delta z)^2$
Convection[a]	$z>0$ $(V>0)$	$V(z_{i-1})/\Delta z$	$V(z_i)/\Delta z$	0
	$z<0$ $(V<0)$	0	$-V(z_i)/\Delta z$	$-V(z_{i+1})/\Delta z$
	p $(\frac{dV}{dz}>0)$	0	$\frac{1}{3}\frac{dV}{dz}\frac{p_i}{P_i^{i+1}}$	$\frac{1}{3}\frac{dV}{dz}\frac{p_{i+1}}{P_i^{i+1}}$
Diffusive reacceleration[a]	p	$-\frac{D_{pp,i}-D_{pp,i-1}}{P_{i-1}^{i\,2}} + \frac{2}{P_{i-1}^i}\left(\frac{D_{pp,i}}{P_{i-1}^{i+1}}+\frac{D_{pp,i-1}}{p_{i-1}}\right)$	$-\frac{D_{pp,i}-D_{pp,i-1}}{P_{i-1}^{i\,2}} + \frac{2D_{pp,i}}{P_{i-1}^{i+1}}\left(\frac{1}{P_i^{i+1}}+\frac{1}{P_{i-1}^i}\right) + \frac{2D_{pp,i}}{P_{i-1}^i p_i}$	$\frac{2D_{pp,i+1}}{P_{i-1}^{i+1}P_i^{i+1}}$
Energy loss[a]	p	0	$-\dot{p}_i/P_i^{i+1}$	$-\dot{p}_{i+1}/P_i^{i+1}$
Fragmentation	R,z,p	0	$1/3\tau_f$	0
Radioactive decay	R,z,p	0	$1/3\tau_r$	0

[a] $P_j^i \equiv p_i - p_j$

Differencing scheme The coefficients of the finite-differencing scheme we use for 2 spatial dimensions are given in Table 1. Since we have a 3-dimensional (R, z, p) problem we use 'operator splitting' to handle the implicit solution, as follows. We apply the implicit updating scheme alternately for the operator in each dimension in turn, keeping the other two coordinates fixed. The source and fragmentation, decay terms are used in every step, so to account for the 3 substeps, $\frac{1}{3}q_i$ and $\frac{1}{3\tau}$ are used instead of q_i, $1/\tau$ for the source term and the fragmentation, decay terms respectively. The coefficients for 3 spatial dimension are the same except that R is replaced by (x,y) and the finite differencing coefficients have the same form as for z, and $\frac{1}{4}q_i$ and $\frac{1}{4\tau}$ are used for the source term and the fragmentation, decay terms respectively, to account for the 4 substeps. With this scheme the solution can be done via the tridiagonal solution for each dimension in turn, as described in Press (1992). The spatial 3D scheme is simpler than the 2D one since the diffusion operator is easier to formulate (x,y,z have the same form), and in addition it does not have the problem of the boundary condition at R=0.

Stability, convergence, accuracy. The method was found to be stable for all α, and this property can be exploited to advantage by starting with $\alpha \gg 1$ (see below). The standard alternating direction implicit (ADI) method, in which the full operator is used to update each dimension implicitly in turn, is more accurate but was found to be unstable for $\alpha > 1$. This is a disadvantage when treating problems with many timescales, but can be used to generate an accurate solution from an approximation generated by the non-ADI method.

A check for convergence is performed by computing the timescale $\frac{\psi}{\partial\psi/\partial t}$ from eq. (46) and requiring that this be large compared to all diffusive and energy loss timescales. The main problem in applying the method in practice is the wide range of time-scales, especially for the electron case, ranging from 10^4 years for energy losses to 10^9 years for diffusion around 1 GeV in a large halo. Use of a time step Δt appropropriate to the smallest time-scales guarantees a reliable solution, but requires a prohibitively large number of steps to reach the long time-scales. The following accelerated technique was found to work well: start with a large Δt appropriate for the longest scales, and iterate until a stable solution is obtained. This solution is then accurate only for cells with $\alpha \ll 1$; for other cells the solution is stable but inaccurate. Then reduce Δt by a factor (0.5 was adopted) and continue the solution. This process is repeated until $\alpha \ll 1$ for all cells, when the solution is accurate everywhere. It is found that the inaccurate parts of the solution quickly decay as soon as the condition $\alpha < 1$ is reached for a cell. As soon as all cells satisfy $\alpha < 1$ the solution is continued with the ADI method to obtain maximum accuracy. A typical run starts with $\Delta t = 10^9$ years and ends with $\Delta t = 10^4$ years for nucleons and 10^2 years for electrons performing ~ 60 iterations per Δt. In this way it is possible to obtain reliable solutions in a reasonable computer resources, although the CPU required is still considerable.

NB the part about continuing with ADI depending on α is not implemented in the current GALPROP version. It was in the f90 version. To be updated.

added 4 Jan 2011): A detailed evaluation of the accuracy of the solutions by both the accelerated and constant-step techniques has been added, see Section 'Tests of GALPROP'. It is recommended to vary the number of repetitions per stepsize in the accelerated method, and to compare with a constant step run a least once as a check on the accuracy for the particular case being run. For the constant step runs, a typical timestep is 1000 years, repeated 10^6 times to get a steady-state solution.

All results are output as FITS datasets for subsequent analysis.

Derivations of differencing formulae. The differencing schemes for the various terms are derived in detail below, with the relevant code in each case.

2D DIFFUSION

$$\frac{\partial \psi}{\partial t} = \vec{\nabla} \cdot (D_{xx} \vec{\nabla} \psi) \tag{33}$$

R term

$$\frac{1}{R}\frac{\partial}{\partial R}\left(R\, D_{xx}\frac{\partial \psi}{\partial R}\right) = \frac{2}{R_i}\frac{D_{xx}}{R_{i+1}-R_{i-1}}\left\{R_{i+1}\frac{\psi_{i+1}-\psi_i}{R_{i+1}-R_i} - R_{i-1}\frac{\psi_i-\psi_{i-1}}{R_i-R_{i-1}}\right\}. \tag{34}$$

Setting $R_{i+1} - R_i = R_i - R_{i-1} = \Delta R$, one can obtain the following expressions in terms of our standard form (eq. [27])

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= D_{xx}\frac{2R_i - \Delta R}{2R_i(\Delta R)^2},\\
\frac{\alpha_2}{\Delta t} &= D_{xx}\frac{2R_i}{R_i(\Delta R)^2},\\
\frac{\alpha_3}{\Delta t} &= D_{xx}\frac{2R_i + \Delta R}{2R_i(\Delta R)^2}.
\end{aligned} \tag{35}$$

z term

$$\frac{\partial}{\partial z}\left(D_{xx}\frac{\partial \psi}{\partial z}\right) = \frac{2D_{xx}}{z_{i+1}-z_{i-1}}\left\{\frac{\psi_{i+1}-\psi_i}{z_{i+1}-z_i} - \frac{\psi_i-\psi_{i-1}}{z_i-z_{i-1}}\right\} = \frac{Dxx}{(\Delta z)^2}\left(\psi_{i+1} - 2\psi_i + \psi_{i-1}\right). \tag{36}$$

Setting $z_{i+1} - z_i = z_i - z_{i-1} = \Delta z$, we obtain the following expressions in terms of our standard form (eq. [27])

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= \frac{D_{xx}}{(\Delta z)^2},\\
\frac{\alpha_2}{\Delta t} &= \frac{2D_{xx}}{(\Delta z)^2},\\
\frac{\alpha_3}{\Delta t} &= \frac{D_{xx}}{(\Delta z)^2}.
\end{aligned} \tag{37}$$

The relevant code in galprop.cc:

```
for (ir = 0; ir < particle.n_rgrid; ++ir) {
for (iz = 0; iz < particle.n_zgrid; ++iz) {
for (ip = 0; ip < particle.n_pgrid; ++ip) {

alpha1_z.d2[ir][iz].s[ip] = particle.Dxx.d2[ir][iz].s[ip]*pow(particle.dz,-2.)
alpha2_r.d2[ir][iz].s[ip] = particle.Dxx.d2[ir][iz].s[ip]*pow(particle.dr,-2.);

if (0 == ir) // use: Dxx/(R[i-1/2] dR)*{Ri(U[i+1]-Ui)-R[i-1](Ui-U[i-1])}; R[i-1]=0
 alpha3_r.d2[ir][iz].s[ip] = alpha2_r.d2[ir][iz].s[ip] *2.;

else { // use: Dxx/(Ri dR)*{R[i+1/2](U[i+1]-Ui)-R[i-1/2](Ui-U[i-1])}

 alpha1_r.d2[ir][iz].s[ip] = alpha2_r.d2[ir][iz].s[ip]*(1. - particle.dr/2./particle.r[ir]);
 alpha3_r.d2[ir][iz].s[ip] = alpha2_r.d2[ir][iz].s[ip]*(1. + particle.dr/2./particle.r[ir]);

}
}}}

alpha2_z = alpha1_z;
alpha2_z *= 2.;
alpha3_z = alpha1_z;

alpha2_r *= 2.;

double factor = dt*pow(kpc2cm, -2.);

alpha1_r *= factor;
alpha1_z *= factor;
alpha2_r *= factor;
alpha2_z *= factor;
alpha3_r *= factor;
alpha3_z *= factor;
```

3D DIFFUSION

This is the same as 2D except that the R-term is replace by x, y, and the coefficients are then exactly analogous to those for z.

x term

$$\begin{aligned}\frac{\alpha_1}{\Delta t} &= \frac{D_{xx}}{(\Delta x)^2},\\ \frac{\alpha_2}{\Delta t} &= \frac{2D_{xx}}{(\Delta x)^2},\\ \frac{\alpha_3}{\Delta t} &= \frac{D_{xx}}{(\Delta x)^2}.\end{aligned} \tag{38}$$

y term

$$\begin{aligned}\frac{\alpha_1}{\Delta t} &= \frac{D_{xx}}{(\Delta y)^2},\\ \frac{\alpha_2}{\Delta t} &= \frac{2D_{xx}}{(\Delta y)^2},\\ \frac{\alpha_3}{\Delta t} &= \frac{D_{xx}}{(\Delta y)^2}.\end{aligned} \tag{39}$$

NB while D_{xx} is assumed here to be isotropic, a generalization to anisotropic diffusion with different D in x, y, z is a simple extension which will be implemented in a future version of GALPROP.

DIFFUSIVE REACCELERATION

In terms of 3-D momentum phase-space density $f(\vec{p})$ the diffusive reacceleration equation is

$$\frac{\partial f(\vec{p})}{\partial t} = \vec{\nabla}_p \cdot [D_{pp} \vec{\nabla}_p f(\vec{p})] = \frac{1}{p^2}\frac{\partial}{\partial p}\left[p^2 D_{pp}\frac{\partial f(p)}{\partial p}\right]. \tag{40}$$

The distribution is assumed isotropic so $f(\vec{p}) = f(p)$ where $p = |\vec{p}|$. First we rewrite the equation in terms of $\psi(p) = 4\pi p^2 f(p)$ instead of $f(p)$ and expand the inner differential:

$$\frac{\partial \psi}{\partial t} = \frac{\partial}{\partial p}\left[p^2 D_{pp}\frac{\partial}{\partial p}\frac{\psi}{p^2}\right] = \frac{\partial}{\partial p} D_{pp}\left[\frac{\partial \psi}{\partial p} - \frac{2\psi}{p}\right]. \tag{41}$$

The differencing scheme is then

$$\frac{2}{p_{i+1}-p_{i-1}}\left[D_{pp,i+1}\left(\frac{\psi_{i+1}-\psi_i}{p_{i+1}-p_i} - \frac{2\psi_{i+1}}{p_{i+1}}\right) - D_{pp,i-1}\left(\frac{\psi_i-\psi_{i-1}}{p_i-p_{i-1}} - \frac{2\psi_{i-1}}{p_{i-1}}\right)\right]. \tag{42}$$

In terms of our standard form (eq. [27]) the coefficients for reacceleration are

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= \frac{2D_{pp,i-1}}{p_{i+1}-p_{i-1}}\left(\frac{1}{p_i-p_{i-1}} + \frac{2}{p_{i-1}}\right), \\
\frac{\alpha_2}{\Delta t} &= \frac{2}{p_{i+1}-p_{i-1}}\left(\frac{D_{pp,i+1}}{p_{i+1}-p_i} + \frac{D_{pp,i-1}}{p_i-p_{i-1}}\right), \\
\frac{\alpha_3}{\Delta t} &= \frac{2D_{pp,i+1}}{p_{i+1}-p_{i-1}}\left(\frac{1}{p_{i+1}-p_i} - \frac{2}{p_{i+1}}\right).
\end{aligned} \tag{43}$$

One more scheme (#2) comes from further detalization

$$\frac{d\psi}{dt} = \frac{\partial D_{pp}}{\partial p}\frac{\partial \psi}{\partial p} + D_{pp}\frac{\partial^2 \psi}{\partial p^2} - 2\frac{\partial}{\partial p}\frac{D_{pp}\psi}{p}. \tag{44}$$

Here it is

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= -\frac{D_{pp,i}-D_{pp,i-1}}{(p_i-p_{i-1})^2} + \frac{2D_{pp,i}}{(p_{i+1}-p_{i-1})(p_i-p_{i-1})} + \frac{2D_{pp,i-1}}{(p_i-p_{i-1})p_{i-1}}; \\
\frac{\alpha_2}{\Delta t} &= -\frac{D_{pp,i}-D_{pp,i-1}}{(p_i-p_{i-1})^2} + \frac{2D_{pp,i}}{p_{i+1}-p_{i-1}}\left(\frac{1}{p_{i+1}-p_i} + \frac{1}{p_i-p_{i-1}}\right) + \frac{2D_{pp,i}}{(p_i-p_{i-1})p_i}; \\
\frac{\alpha_3}{\Delta t} &= \frac{2D_{pp,i+1}}{(p_{i+1}-p_{i-1})(p_{i+1}-p_i)}.
\end{aligned} \tag{45}$$

This scheme is used in `propel.cc`. The relevant part of the code is as follows:

```
   for (ir = 0; ir < particle.n_rgrid; ++ir) {
   for (iz = 0; iz < particle.n_zgrid; ++iz) {
   for (ip = 1; ip < particle.n_pgrid-1; ++ip) {

 // alternative scheme #2, most detailed
 alpha1_p.d2[ir][iz].s[ip] +=
   (-(particle.Dpp.d2[ir][iz].s[ip] - particle.Dpp.d2[ir][iz].s[ip-1])
    /(particle.p[ip] - particle.p[ip-1])
    + 2.*particle.Dpp.d2[ir][iz].s[ip]
    /(particle.p[ip+1] - particle.p[ip-1])
```

```
       + 2.*particle.Dpp.d2[ir][iz].s[ip-1]/particle.p[ip-1])
      /(particle.p[ip]-particle.p[ip-1]);

    alpha2_p.d2[ir][iz].s[ip] +=
      -(particle.Dpp.d2[ir][iz].s[ip]-particle.Dpp.d2[ir][iz].s[ip-1])
      /pow(particle.p[ip] - particle.p[ip-1], 2.)
      + 2.*particle.Dpp.d2[ir][iz].s[ip]
      /(particle.p[ip+1] - particle.p[ip-1])
      *(1./(particle.p[ip+1] - particle.p[ip])
+ 1./(particle.p[ip] - particle.p[ip-1]))
      + 2.*particle.Dpp.d2[ir][iz].s[ip]
      /(particle.p[ip] - particle.p[ip-1])
      /particle.p[ip];

    alpha3_p.d2[ir][iz].s[ip] +=
      2.*particle.Dpp.d2[ir][iz].s[ip]
      /(particle.p[ip+1] - particle.p[ip-1])
      /(particle.p[ip+1] - particle.p[ip]);
```

CONVECTION

Spatial part:

$$\frac{\partial \psi}{\partial t} = \vec{\nabla} \cdot (-\vec{V}\psi) \tag{46}$$

The wind is assumed to blow outwards from the disc, so V is +ve for $z > 0$, -ve for $z < 0$. For the differencing we use the physical fact that for convection ψ can only depend on upstream values, so gridpoint i depends on $i, i-1$ for $z > 0$, and on $i, i+1$ for $z < 0$ (upstream differencing).

$z > 0$: $-\frac{V_i\psi_i - V_{i-1}\psi_{i-1}}{\Delta z}$

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= \frac{V_{i-1}}{\Delta z} \\
\frac{\alpha_2}{\Delta t} &= \frac{V_i}{\Delta z} \\
\frac{\alpha_3}{\Delta t} &= 0
\end{aligned} \tag{47}$$

$z < 0$: $-\frac{V_{i+1}\psi_{i+1} - V_i\psi_i}{\Delta z}$

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= 0 \\
\frac{\alpha_2}{\Delta t} &= -\frac{V_i}{\Delta z} \\
\frac{\alpha_3}{\Delta t} &= -\frac{V_{i+1}}{\Delta z}
\end{aligned} \tag{48}$$

Since V is +ve for $z > 0$, -ve for $z < 0$, so in propel.cc the spatial convection α's are all finally +ve.

The relevant code in propel.cc : (V in km s^{-1}, $\frac{dV}{dz}$ in km s^{-1} kpc^{-1})

```
for (iz = 0; iz < particle.n_zgrid; ++iz) { // numerator = abs convection velocity in cm s^-1
 double az1 = (particle.z[iz] > 0.) ?
  (galdef.v0_conv + galdef.dvdz_conv*fabs(particle.z[iz-1]))*1.e5/particle.dz*dt/kpc2cm: 0.;
 double az2 =
```

```
  (galdef.v0_conv + galdef.dvdz_conv*fabs(particle.z[iz]  ))*1.e5/particle.dz*dt/kpc2cm;
 double az3 = (particle.z[iz] < 0.) ?
  (galdef.v0_conv + galdef.dvdz_conv*fabs(particle.z[iz+1]))*1.e5/particle.dz*dt/kpc2cm: 0.;
```

Momentum part:

$$\frac{\partial\psi}{\partial t} = \frac{\partial}{\partial p}\frac{p}{3}(\vec{\nabla}\cdot\vec{V})\psi \tag{49}$$

Since we consider only adiabatic energy losses ($dV/dz > 0$), physically energy bin i depends on higher energies only, so we consider only $i, i+1$ (upstream differencing). We consider here only $\frac{dV}{dz}$= constant > 0 (note this holds for all z, negative as well as positive).

$$\frac{1}{3}\frac{dV}{dz}\frac{p_{i+1}\psi_{i+1} - p_i\psi_i}{p_{i+1} - p_i}$$

.

$$\begin{aligned}
\frac{\alpha_1}{\Delta t} &= 0 \\
\frac{\alpha_2}{\Delta t} &= \frac{1}{3}\frac{dV}{dz}\frac{p_i}{p_{i+1} - p_i} \\
\frac{\alpha_3}{\Delta t} &= \frac{1}{3}\frac{dV}{dz}\frac{p_{i+1}}{p_{i+1} - p_i}
\end{aligned} \tag{50}$$

The relevant code in propel.cc : (V in km s^{-1}, $\frac{dV}{dz}$ in km s^{-1} kpc^{-1})

```
double ap2 = particle.p[ip]  /3.*galdef.dvdz_conv/(particle.p[ip+1]-particle.p[ip])*1.e5/kpc2cm;
double ap3 = particle.p[ip+1]/3.*galdef.dvdz_conv/(particle.p[ip+1]-particle.p[ip])*1.e5/kpc2cm;
```

5.5.4 nuclei_normalize & electrons_normalize

Normalizes all nuclei to proton flux from galdef file. Applied at end of all nuclei processing but before computation gamma-rays (which use p, He and electrons). The value of E_{kin} is taken as the reference value for the proton normalization since this is already available in the *galdef* file.

Method: compute the proton flux $\frac{c}{4\pi}n(p)$ at the reference E_{kin} at $R = R_0$ and $z = 0$ by interpolation, and hence obtain the normalizing factor to get the correct value as specified in the *galdef* file. Renormalizes the proton and all other nuclei fluxes by this same factor. The units of all spectra are then $\frac{c}{4\pi}n(p)$. Note that the source abundances are already taken into account in *create_transport_arrays*.

5.6 store_gcr & store_gcr_full

5.7 Synchrotron radiation

The routines involved are `gen_synch_emiss, synchrotron_emissivity_B_field, synchrotron_emissivity, gen_synch_skymap`.

These compute the synchrotron emissivity including Stokes Q and U, and generates skymaps from these for total and Stokes Q and U.

`synchrotron_emissivity.cc` is the basic physics routine computing the emissivity using Bessel functions, for a given B-field (projected onto perpendicular to the line-of-sight) and random B-field. Total emissivity and Stokes Q and U emissivities are computed, for a given synchrotron frequency and electron/positron energy. The formulae in the literature are somewhat confusing using various conventions, and a unifying survey is contained in this routine, which to the best of our knowledge is correctly coded.

5.7.1 Regular field

The synchrotron emissivity of an isotropic distribution of monoenergetic relativistic particles in a uniform magnetic field has polarized components parallel and perpendicular to the projection of the field on the line-of-sight to the observer (Longair: High Energy Astrophysics, Rybicki and Lightman: Radiative Processes in Astrophysics)

$$\epsilon_{\perp}(\nu) = \frac{\sqrt{3}}{2}\ \frac{e^3}{mc^2} B_{perp}\ [F(x) + G(x)] \tag{51}$$

$$\epsilon_{\parallel}(\nu) = \frac{\sqrt{3}}{2}\ \frac{e^3}{mc^2} B_{perp}\ [F(x) - G(x)] \tag{52}$$

where x = ν/ν_c, with $\nu_c = \frac{3}{4\pi}\ \frac{e}{mc} B_{perp}\gamma^2$ and with γ the electron Lorentz factor, and B_{perp} is the projection of **B** on the plane perpendicular to the line-of-sight. The functions F(x) and G(x) are defined in terms of Bessel functions (Longair: High Energy Astrophysics, Rybicki and Lightman: Radiative Processes in Astrophysics) with:

$$F(x) = x\ \int_x^{\mathrm{inf}}\ K_{5/3}(x')\ dx'$$
$$G(x) = x\ K_{2/3}(x) \tag{53}$$

where $K_{5/3}(x)$ and $K_{2/3}(x)$ are the modified Bessel functions of order 5/3 and 2/3. They are conveniently provided as C library functions in the GNU Scientific Library[6], and this implementation is used by GALPROP. The resulting synchrotron spectrum has a broad maximum centred roughly at the frequency ν_c and the maximum has a value ν_{max}=0.29 ν_c

The total emissivity is given by the sum of the two terms above:

$$\epsilon(\nu,\gamma) = \sqrt{3}\ \frac{e^3}{mc^2} B_{perp}\ F(x) \tag{54}$$

5.7.2 Random field

For a randomly oriented field the emissivity is isotropic and obtained by integrating the regular field expressions over all solid angles. The result is given by (Ghisellini et al. 1988 ApJ 334,5)

$$\epsilon_{rand}(\nu) = C\ x^2 [K_{4/3}K_{1/3} - \frac{3}{5} x (K_{4/3}K_{4/3} - K_{1/3}K_{1/3})] \tag{55}$$

$x = \nu/\nu_c$, $\nu_c = \frac{3}{2\pi}\frac{e}{mc} B_{ran}\gamma^2$, $C = 2\sqrt{3}\frac{e^3}{mc^2} B_{ran}$ erg s^{-1} Hz^{-1}, and the Bessel functions $K_{4/3}, K_{1/3}$ are again computed using the GNU Scientific Library.

In the GALPROP implementation, the code has been checked by integrating the regular field expression over solid angle, giving exact agreement with this formula.

5.7.3 `synchrotron_emissivity_B_field.cc`

uses the selected B-field model to invoke `synchrotron_emissivity.cc` for a particular position, synchrotron frequency and electron/positron energy, and computes the Stokes parameters in a uniform reference system suitable for line-of-sight integration.

For the regular field, the emissivities perpendicular and parallel to the projection of **B** onto the line-of-sight from the observer to the emitting volume element $\epsilon_{\perp}, \epsilon_{\parallel}$ are given above.

We follow Waelkens etal 2009 (A&A 495, 697) description of the Hammurabi code for the detailed formulae for Stokes parameters.

The emissivities for Stokes parameters I, Q, U are then defined by

$$I = \epsilon_{\perp} + \epsilon_{\parallel}$$

[6]http://www.gnu.org/software/gsl

$$P = \epsilon_\perp - \epsilon_\parallel$$

$$Q = P\cos(2\chi)$$
$$U = P\sin(2\chi)$$

I is the total intensity, $P = \sqrt(Q^2 + U^2)$ is the polarized intensity, $I - P = 2\epsilon_\parallel$ is the unpolarized intensity. The polarized fraction is $\frac{P}{I}$. Unpolarized radiation has $Q = U = 0$, totally polarized has $I = P$.

$$\chi = \frac{1}{2}\arctan(U/Q)$$

χ is the angle between the polarization direction of the electric vector and the Galactic longitude meridian (N-S) (convention as in Page etal. 2007 ApJ 170, 335.) The projection of **B** on the plane perpendicular to the line-of-sight then has direction $\chi + \pi/2$ since the emission is polarized perpendicular to the projection **B** onto the line-of-sight (NB $\epsilon_\perp > \epsilon_\parallel$ for synchrotron radiation). With this definition of χ, **B** parallel to the Galactic plane has $\chi = 0, Q > 0, U = 0$, while **B** pointing towards the North Galactic Pole has $\chi = \pi/2, Q < 0, U = 0$. **B** at $\pi/4$ to the meridian has $\chi = \pi/4, Q = 0, U > 0$, and at $-\pi/4$ to the meridian $\chi = -\pi/4, Q = 0, U < 0$.

This routine uses direction cosines to perform the necessary angular calculations.

Setting the galdef parameter verbose = -1100 produces a detailed output of the calculation at every point, useful for testing.

Note that the formulation here is more general than that often used which gives the Stokes parameters directly in terms of the components of **B** (e.g. Page et al. 2007 WMAP, Ensslin et al. 2009 Astr. Nachrichten 327,626); the latter is only possible under the assumption of an electron spectral index of 3, while ours (and Waelkens etal., Hammurabi) is for any electron spectrum. The simplified formulae are useful for understanding the relation between the topology of **B** and the Stokes parameters, and for checking the code. These relations are of the kind

$$I \propto \int (B_x^2 + B_z^2) ds$$

$$Q \propto \int (B_x^2 - B_z^2) ds$$

$$U \propto \int (2B_x B_z) ds$$

where B_x, B_z are projected on the plane of the sky to the observer, s is the line-of-sight.

5.7.4 `synchrotron_emissivity_aws.cc`

provides a convenient interface to invoke `synchrotron_emissivity.cc` and `synchrotron_emissivity_B_field.cc`.

`gen_synch_emiss.cc` uses the above routines to generate total, Q and U emissivities for the whole 2D or 3D grid, for all required frequencies. The integration over electron (primary and secondary) and positron spectra is done here.

`gen_synch_skymap` integrates the emissivities (total, Q, U) over the line-of-sight to produce the corresponding synchrotron skymaps. The integration step is taken from GALDEF parameter LoS_step. With GALPROP calculation of emissivity on the grid, we integrate over the line-of-sight to get the synchrotron intensity for the regular and random fields. The synchrotron intensity at frequency ν is then given by

$$I(\nu) = \int \epsilon(\nu)\ ds \tag{56}$$

The observed brightness of the radiation seen in a given direction is

$$T(\nu) \propto \frac{c^2 I(\nu)}{2\ \nu^2} \tag{57}$$

5.7.5 Free-free absorption and emission

Free-free absorption by ionized hydrogen (WIM: warm interstellar medium) is important at low radio frequencies (below 100 MHz in the plane, below 10 MHz at high latitudes). This is now implemented (*r951*). It is controlled by the electron temperature T$_e$ and the clumping factor since absorption depends on the square of the electron density. The clumping factor is related to the filling factor; for equal-size clouds, clumping factor = 1/filling factor. Clumping factors of 10-100 are typical based on a variety of data (pulsar DM, Hα EM, free-free emission, synchrotron absorption). Free-free emission can also be computed, with a special option to replace the synchrotron skymaps with free-free skymaps, using the same frequency grid as for synchrotron. The free-free skymap is always output to a separate file when synchrotron is selected ($\mathring{9}79$). The free-free emission is subject to absorption using the same scheme. The free electron density uses the model described in `nHII.cc`. Controlled by parameters free_free_absorption, HII_Te, HII_clumping_factor. Future developments will include spatial dependence of these quantities, which are known to vary in the Galaxy, and a more sophisticated model of the WIM. *Full details required.*

5.8 `gen_bremss_emiss, gen_IC_emiss, & gen_pi0_emiss`

π^o-decay emissivity from CR protons and Helium on interstellar hydrogen and Helium.

Units:(H atom)$^{-1}$ sr^{-1} s^{-1} MeV^{-1}.

IC emissivity from CR primary electrons and secondary electrons and positrons.

Units: cm^{-3} sr^{-1} s^{-1} MeV^{-1}.

The integral over particles is over log(E_{kin}), see section General Principles. This explains the formula used in this routine. The factor includes GeV^{-1} to MeV^{-1} and barn to cm^2. A correction for the Helium abundance of the gas as in the galdef file is applied in computing the emissivities. Helium nuclei in CR are explicitly included here.

5.9 `gen_bremss_skymap, gen_IC_skymap, & gen_pi0_skymap`

π^o-decay, bremsstrahlung and IC skymaps as function of (l, b, E_γ).

Units: internal: cm^{-2} sr^{-1} s^{-1} MeV^{-1}, for output: MeV^2 cm^{-2} sr^{-1} s^{-1} MeV^{-1}.

The skymaps are produced by integrating the emissivities over the line-of-sight for an observer at the Solar position, for each (l, b) direction in the map. For IC this is straightforward since the volume emissivity is directly calculated using the ISRF and electron, positron spectra.

For bremsstrahlung and π^o-decay the emissivity is per nucleon of gas. To best implement observed Galactic structure in the gas, HI and CO radio-astronomical surveys in Galacticentric rings are used together with the user-defined H$_2$-to-CO relation given in the *galdef* file. These data only give column densities per ring, so that the variation of emissivity and gas density *within* each ring has to be taken into account by some approximation. This is done using a gas-density model as a function of (R, z) as follows:

$$I\gamma = \Sigma_i \frac{N_{HI,i} + 2X_{CO,i}W_{CO,i}}{\int_{ring\ i}(n_{HI} + 2n_{H_2})ds} \times \int_{ring\ i} q_\gamma(n_{HI} + 2n_{H_2})ds$$

where i indexes the Galactocentric rings. Here $(n_{HI} + 2n_{H_2})$ is the model gas density at any point (R, z) as described in Section 2.1.1, s is the line-of-sight distance, $(N_{HI,i} + 2X_{CO,i}W_{CO,i})$ is the survey-based column density in ring i. The integral is thus corrected for the observed column density while maintaining the model-based variation within the ring. The integrals are performed with a resolution in s of 10 pc.

6 Point sources of CR: spatial and temporal aspects

A new development from version 4 is the inclusion of spatial and temporal variations, which can be followed explicitly in the 3D case. Supernova remnants are produced at some rate (e.g. 1/100 yr) and produced cosmic rays for some time (e.g. 10,000 yr). The first step is to include point sources at arbitrary positions specified by the user, by this is mainly useful for testing and does not address the problem of fluctuations.

The correct approach is to consider the source function as $f(x, y, z, t)$. The problem is that the solution has to be followed on a time scale comparable to the SN rate, i.e. 100 years or less. This would not allow the advantages of the equilibrium solution scheme used in `galprop`. However provided the equilibrium solution is first derived, the short time steps can be started afterwards and will lead to the correct "recent" history of the system.

To model the source function we need an explicit $f(x, y, z, t)$ which has the right properties of producing short-lived events at different positions. This can be done by defining first a regular grid on which SNR can occur, and then generating a random phase for each grid point. The function of time at each point is then a suitable function with the required duty cycle and period.

The basic spatial grid itself can be used, there is no need to define a separate one for the sources. The required parameters are: the probability per unit time and volume of a SNR events, and the time for which it is active. To express both as a time, define `SNR_time_interval` as the time between events in a kpc^3 volume in the Solar vicinity for the first parameter.

The structure Galaxy contains an array giving the phase and rate for each cell. The phases are chosen randomly at the start of the program. The rates are determined from the time interval and the source distribution which is taken as the same as that defined for constant sources. At any time, the on/off state of an SNR in a cell is determined from the time relative to the phase, the time between SNR in the cell (`SNR_cell_time`) and the live time.

The propagation code (`propel.cc`) is divided into two sections to handle the smooth source function using the decreasing timesteps, followed by a section with constant timesteps to handle the SNR events on a fine timescale. These are called `timestep_mode`=1, 2 respectively. The number of timesteps in each mode can be specified independently, but in mode 2 the shortest timestep of mode 1 is used.

If the number of timesteps in mode 1 is set to zero then only mode 2 is used. For a combination of modes it is necessary to normalize the source functions to be consistent: this has to be done by adjusting the input per SNR (`Q_SNR`) since the rate and live time are given. The source function returned by `source_distribution.cc` is defined as production per unit volume (cm^3) per second. (it gets multiplied by spectral and abundance factors later).

The assigned source function in source_SNR_event has to satisfy

$$source\_function * SNR\_livetime/SNR\_cell\_time = source\_distribution(x, y, z)$$

so

$$source\_function = source\_distribution(x, y, z)/SNR\_livetime * SNR\_cell\_time$$

and the SNR input when on should be always same everywhere by construction.
`create_SNR.cc` uses

$$SNR\_cell\_time =$$

$$SNR\_interval * source\_distribution(Sun)/source\_distribution(x, y, z)/cellvolume$$

so the formula used by `source_SNR_event.cc` is

$$source\_function = source\_distribution(x, y, z)/SNR\_livetime * SNR\_interval$$

$$*source\_distribution(Sun)/source\_distribution(x, y, z)/cellvolume$$

which simplifies to

$$source\_function = SNR\_interval/SNR\_livetime * source\_distribution(Sun)/cellvolume$$

and the output rate of an SNR =

$$source\_function * cellvolume = SNR\_interval/SNR\_livetime * source\_distribution(Sun)$$

which is a constant as required. Note that $SNR\_interval$ is for 1 kpc^3 and so has units of years * kpc^3 while $source\_distribution(Sun)$ has units of cm^{-3} s^{-1} so the SNR output rate units are s^{-1} as required. Similarly $source\_function$ has the same units as $source\_distribution$, as required.

7 Analytical solutions

Analytical solutions can be obtained for simple cases which provide an essential test of the correct working of *galprop*.

7.1 Simple diffusion equation without losses

The propagation equation for the particle density ψ in this case is:

$$\frac{\partial \psi}{\partial t} = q(\vec{r}, p) + \vec{\nabla} \cdot D_{xx} \vec{\nabla} \psi \tag{58}$$

Consider the 1-D case with boundary condition $\psi(z_h) = \psi(-z_h) = 0$ and $q = 0$ except at $z = 0$.
The steady-state 1-D form is

$$q(z) = D_{xx} \frac{d^2 \psi}{dz^2} \tag{59}$$

The solution is

$$\psi = A(z_h - |z|) \tag{60}$$

The constant A is determined from $\int q(z)dz = D_{xx}[\frac{d\psi}{dz}]_-^+ = 2AD_{xx}$ so the solution for primaries is

$$\psi_p = \frac{Q}{2D_{xx}}(z_h - |z|) \tag{61}$$

where $Q = \int q(z)dz$ is the surface CR emissivity. The solution is just the expression of the conservation of particles, hence the equality of the source term with the the diffusive CR current $-D_{xx}\frac{d\psi}{dz}$, with half the current going in each z-direction.

Secondary production just replaces the source term q by the spallation term $\psi_p(0)n\sigma\beta c$ where σ is the production cross-section and n the gas density:

$$\psi_s = \frac{\psi_p(0)n\sigma\beta c\Delta z}{2D_{xx}}(z_h - |z|) \tag{62}$$

so the secondary/primary ratio in the disk is

$$\psi_s(0)/\psi_p(0) = \frac{n\Delta z\sigma\beta c z_h}{2D_{xx}} \tag{63}$$

This ignores the destruction terms. They can be included by replacing q by $q - n\sigma_p\beta c\psi_p(0)$ giving

$$\psi_p = \frac{Q(z_h - |z|)}{2D_{xx} + n\Delta z\sigma_p\beta c z_h} \tag{64}$$

and correspondingly

$$\psi_s = \frac{\psi_p(0)n\sigma\beta c\Delta z}{2D_{xx} + n\Delta z\sigma_s\beta c z_h}(z_h - |z|) \tag{65}$$

where σ_p, σ_s are the primary and secondary destruction cross-sections.

The secondary/primary ratio in the disk is

$$\psi_s(0)/\psi_p(0) = \frac{n\Delta z\sigma\beta c z_h}{2D_{xx} + n\Delta z\sigma_s\beta c z_h} \tag{66}$$

7.2 Simple diffusion equation without losses, with decay

The propagation equation for the particle density ψ in this case is:

$$\frac{\partial\psi}{\partial t} = q(\vec{r},p) + \vec{\nabla}\cdot D_{xx}\vec{\nabla}\psi - \frac{\psi}{\tau_r} \quad (67)$$

Consider the 1-D case with boundary condition $\psi(z_h) = \psi(-z_h) = 0$ and $q = 0$ except at $z = 0$.
The steady-state 1-D form is

$$q(z) = D_{xx}\frac{d^2\psi}{dz^2} - \frac{\psi}{\tau_r} \quad (68)$$

Outside the source region $q(z) = 0$

$$D_{xx}\frac{d^2\psi}{dz^2} - \frac{\psi}{\tau_r} = 0 \quad (69)$$

The general solution is of the form $\psi = Ae^{-kz} + Be^{+kz}$ where $k = \frac{1}{\sqrt{(D_{xx}\tau_r)}}$. The boundary conditions give $Ae^{-kz_h} + Be^{+kz_h} = 0$ and $Ae^{-kz_h} + Be^{+kz_h} = 0$ (so A/B different for $z > 0, z < 0$). For $z > 0$, $B = -Ae^{-2kz_h}$ so $\psi = A(e^{-kz} - e^{-2kz_h}e^{+kz}) = Ae^{-kz}(1 - e^{-2kz_h}e^{+2kz}) = Ae^{-kz}(1 - e^{2k(z-z_h)})$.
The constant A is determined from

$$\int q(z)dz = D_{xx}[\frac{d\psi}{dz}]^+_- = 2D_{xx}[\frac{d\psi}{dz}]^+_0 = 2A[(-ke^{-kz} - ke^{kz})]_{z=0} = -4AkD_{xx}$$

(should not be negative, formula for flux?) so $A = \frac{Q}{4kD_{xx}}$ giving the solution for primaries

$$\psi_p = \frac{Q}{4kD_{xx}}e^{-kz}(1 - e^{2k(z-z_h)}) \quad (70)$$

where $Q = \int q(z)dz$ is the surface CR emissivity.
For $\tau \to \inf$, $k \to 0$ the solution tends to $\psi = \frac{Q}{4kD_{xx}}(1-kz)(2k(z-z_h)) = \frac{Q}{4kD_{xx}}(2k(z-z_h)+2k^2z(z-z_h)) \to \frac{Q}{2D_{xx}}(z - z_h)$ keeping only terms to $O(k)$. (NB sign, should be $z_h - z$). This recovers the linear form for the stable nuclei case above.
We are intererested in e.g. ^{10}Be/^{9}Be i.e. unstable/stable secondaries at $z = 0$.
$\frac{\psi_{su}}{\psi_{ss}} = \frac{Q_{su}}{4kD_{xx}}(1 - e^{-2kz_h})/\frac{Q_{ss}}{2D_{xx}}(z_h - z) = \frac{\sigma_{su}}{\sigma_{ss}}\frac{(1-e^{-2kz_h})}{2kz_h}$

$$\frac{\psi_{su}}{\psi_{ss}} = \frac{\sigma_{su}}{\sigma_{ss}}\frac{(1 - e^{-2z_h/\sqrt{(D_{xx}\tau_r)}})}{2z_h/\sqrt{(D_{xx}\tau_r)}}$$

where $\frac{\sigma_{su}}{\sigma_{ss}}$ is the ratio of production cross-sections for unstable and stable secondaries from the same primary.
For $\tau_r \to 0$ i.e. rapid decay, the exponential $\to 0$ and

$$\frac{\psi_{su}}{\psi_{ss}} = \frac{\sigma_{su}}{\sigma_{ss}}\frac{\sqrt{(D_{xx}\tau_r)}}{2z_h}$$

This can be interpreted as the diffusion distance $\sqrt{(D_{xx}\tau_r)}$ before decay compared with the halo height z_h, giving the ratio of decaying to non-decaying particles.
This shows how the combination of secondary/primary which constrain $\frac{z_h}{D_{xx}}$ and unstable/stable secondaries which constrain $\frac{z_h}{\sqrt{D_{xx}}}$ together allow both D_{xx} and z_h to be determined.

7.3 Simple convection equation without losses

The propagation equation for the particle density ψ in this case is:

$$\frac{\partial\psi}{\partial t} = q(\vec{r},p) - \vec{\nabla}\cdot(\vec{V}\psi) \quad (71)$$

As before we assume $q = 0$ except at $z = 0$. In the steady-state this is just $\vec{\nabla}\cdot(\vec{V}\psi) = 0$ except at z=0 where the source is located. For convection only in the z-direction, this becomes $V\psi$=constant, i.e. just conservation of particles in a uniform flow. Then $V\psi = q$ for q per surface area. So $\psi \propto \frac{1}{V}$, and so if $V \propto z$, $\psi \propto \frac{1}{z}$. For secondaries,

$$V\psi_s = q_s = n\Delta z\psi_p\sigma\beta c$$

so the secondary/primary ratio in the disk is

$$\psi_s(0)/\psi_p(0) = \frac{n\Delta z\sigma\beta c}{V} \quad (72)$$

An order-of-magnitude estimate of the values: $n\Delta z = 10^{21}$ atoms cm^{-2}, σ_{BC}= 100 mb, V=100 km s^{-1} giving $B/C = 0.3$, of the right order. This indicates that quite high wind velocities are required to compete with diffusion. This estimate is valid for V=constant, otherwise it is not clear at which z it should be taken. A better solution for $V(z)$ will be added in future.

8 Enhancements and changes to the code

8.1 Current version end 2010

A description of some of the enhancements can be found in Strong et al. (2009); Vladimirov et al. (2010) from which the following is taken:

* Shared-memory parallel support with OpenMP to take advantage of multi-processor machines
* Memory usage optimization
* Implementation of the HEALPix output of gamma-ray and synchrotron skymaps. The HEALPix format is a standard for radioastronomy applications, as well as for such instruments as WMAP, Planck etc.
* Implementation of the MapCube output for compatibility with Fermi-LAT Science Tools software
* Implementation of gamma-ray skymaps output in Galactocentric rings to facilitate spatial analysis of the Galactic diffuse gamma-ray emission
* More accurate line-of-sight integration for computing diffuse emission skymaps
* 3D modeling of the Galactic magnetic field, both regular and random components, with a range of models from the literature, extensible to any new model as required
* Calculations of synchrotron skymaps on a frequency grid, using both regular and random magnetic fields
* Improved gas maps, which are computed using recent H I and CO (H2 tracer) surveys, with more precise assignment to Galactocentric rings
* A new calculation of the Galactic interstellar radiation field using the FRaNKIE code (Fast Radiation transport Numerical Kode for Interstellar Emission, as described in) and implementation of the corresponding changes in GALPROP
* Considerably increased efficiency of anisotropic inverse Compton scattering calculations
* GALPROP code is compiled to a library for easy linking with other codes (e.g. DarkSUSY , SuperBayeS)
* Numerous bug fixes and code-style improvements
* Improved configuration management via the GNU autotools. Multiple *NIX system and compiler targets (gcc, intel, llvm, open64) are supported
* Bugzilla available at the GALPROP WebRun URL for user-submitted bug tracking

8.2 Version 54

This version replaces Version 50p, and has many enhancements and corrections.

Visible to users:

* Many new parameters, see the parameters description.
* HI, CO surveys specified by file names.
* Healpix and Fermi-LAT Science Tools compatible skymaps output.
* etc. to be continued

Internal changes:
* etc

8.3 Version 50

This version replaced Version 42.3p, and had many enhancements and corrections.

Visible to users:
* Xco(R) user-defined in GALDEF file.
* Output of components from HI, CO separately, in Rings.
* ISRF specified via filename
* high energy IC now OK (was affected by float rounding error)
* synchrotron corrected (factor c was missing)
* wave damping

Internal changes:
* "extern" global variables eliminated, instead placed in Class Galprop. (= better C++ style)
* all functions are members of Class Galprop (except a few generic routines).
* program structure otherwise identical
* all double instead of float, including Class Distribution
* runs on 64-bit machines so will handle larger cases

9 To do list

9.1 Documentation

1. More details on output data (energy scales, projection, etc).
2. Bug list
3. Feedback/user forum.
4. List of papers using *galprop*
5. Platforms e.g. 64-bit
6. Sample galdef, output data, plots for e.g. 3 useful cases.
7. Summary of galdef parameters for standard published models with galdef files.

9.2 Code

1. Improve FITS headers, e.g. at present they do not have the units.
2. Output synchrotron emissivities.
3. Work towards all c++ and eliminate fortran.

10 Program architecture

This new section provides an overview of the structure and processing of the *galprop* package.

10.1 Classes

The main classes are:

Galprop: encapsulates the entire processing. Can be instantiated as required. Advantage is sharing of all data within one class, no global variables exist. All the other classes are instantiated within the Galprop class.

Galaxy: contains all the structures describing the Galaxy used in the program.

Galdef: the parameters used by *galprop*, as entered by the user via the GALDEF parameter file.

Distribution: general purpose array of objects of type Spectrum, in 2D or 3D. Used throught the program.

Spectrum: a 1D array for holding a spectrum (or any other quantity)

Particle: contains all the information about particular species of particle (mass, Z, A etc) and all the arrays required to propagate it.

Skymap: used for all skymaps generated, and has general-purpose output method.

GalacticRadiationField: handles all aspects of the Galactic interstellar radiation field

Configure: used to configure the program for the input/output data directories, and the directory containing GALDEF parameter files.

GCR_data: handles a database of cosmic-ray data with access routines. Not used by the standard main program, but may be used if required by other programs.

10.2 The program

Main program galprop.cc: instantiates Galprop class, runs it, deletes it, and exits.

10.3 Processing

The processing can be described schematically as follows:

```
Read GALDEF file using method of Galdef class.

Create a Galaxy object and initialize it.

Compute all Galaxy arrays for CR particles  as specified in GALDEF parameters
|  Arrays for x,y,z,r,  p, Ekin, dp/dt, fragmentation, decay, Dxx, Dpp
|  ISRF, magnetic field, cross-sections .....


Loop over network iterations
|
| Loop over particle species starting from most massive
| |
| | Propagate particle species
| | | Compute source function as primary and as secondaries from all heavier species
| | |
| | | Time steps decreasing as specified in GALDEF parameters
| |



Output CR distributions

Generate gamma-ray emissivities
Output gamma-ray emissivities

Generate gamma-ray skymaps
Output gamma-ray skymaps

Generate synchrotron emissivities
Output  synchrotron emissivities

Generate synchrotron skymaps
Output synchrotron skymaps

Delete everything

Exit
```

11 Tests of GALPROP

11.1 Propagation algorithm

The technique has been described in Section 5.5.3. Here we test the accuracy of the method comparing the constant small step (CSS) and accelerated solutions. The routine propel_diagnostics.cc gives information during a run, while various solutions can also be compared for complete runs. The first question is how accurate the CSS method is since this is the benchmark method. Here we assume that the coefficients of the propagation are coded correctly and seek to check only whether the steady-state solution is correctly computed. The benchmark case here is electrons, $10^3 - 10^6$ MeV, energy factor 1.2, with a 4 kpc halo and standard diffusion coefficient and reacceleration paratmer. For timesteps of 1000 yr the timescale is below all energy-loss timescales so that 10^6 steps will give the solution for 10^9 years which suffices for a 4 kpc halo (using $z \approx \sqrt{(D_{xx}t)}$). Note that physically the steady-state is an approximation since the injection of CR will certainly not be constant over 10^9 years. This should be remembered when evaluating the results, but in any case we want to be sure that the mathematical solution to a given model is correctly obtained independent of its physical validity.

Solutions are compared by computing the fractional difference for all positions and energies, outputting the result in FITS.

The basic finding of these tests are:

1. The CSS solution: the diagnostic minimum timescale based on the formula for $d\psi/dt$ in terms of the α's given by propel_diagnostics.cc reaches a constant value for any given timestep, while the CR density continues to increase. Hence this diagnostic is apparently not sufficient. Continuing the solution until the CR density is constant leads to an apparently accurate solution after 10^6 steps of 1000 yr. The diagnostic timescale reached based on the formula for $d\psi/dt$ in terms of the α's given by propel_diagnostics.cc decreases with increasing stepsize, i.e. the solutions become less reliable according to this diagnostic; for example a timestep of 1000 yr reaches a constant mininum timescale 10^5 yr after about 10000 steps(check), which would not indicate an accurate solution. Nevertheless continuing the solution for the nominal 10^6 reaches a time-independent CR density and hence apparently a reliable solution, but this is not necessarily the case (see next paragraph). An alternative diagnostic provided by propel_diagnostics.cc uses the maximum fractional change since the last call of this routine, combined with the number of steps elapsed and the timestep; this gives a steadily increasing timescales, and reaches very large values ($> 10^{18}$ years) after 6 10^5 steps of 1000 years for the benchmark case and then goes to machine infinity (i.e. $d\psi/dt = 0$ everywhere) at 8.4 10^5 steps, indicating an accurate solution within machine accuracy. This can be used as a criterion for terminating a timestep loop, since further steps at this timestep value give no change in the solution. This is implemented at r722 via the galdef parameter *solution_convergence* (see parameter descriptions).

The reason for the devation of the differencing diagnostic from the α-based estimate lies in the ADI method used to handle our multi-dimensional problem: the α-based diagnostic uses the full operator for all dimensions, while our adopted ADI method applies the operator for each dimension in turn, which leads to a slightly different solution; using small enough timesteps the operators are sufficiently well mixed to give a good solution by the α-based diagnostic (the timescale becomes very large). It has been verified that for small timesteps the two timescales are consistent, i.e. that the method works correctly in this case and is coded correctly. However this means that the differencing diagnostic is not a completely reliable indicator of a converged solution. To obtain a good solution according to reliable α-based diagnostic requires a very long CSS run, but can also be achieved with the accelerated method provided enough steps per timestep are used and that the final timestep is small enough. More specific examples of this aspect will be added.

This suggests other strategies for the future: an explicit method can be used with the full operator for small enough timesteps to ensure stability, and this could be appended to the solution at the end of an accelerated run. This would avoid the ADI problem described above. Non-ADI methods solving the full system of equations are expensive since the matrix is no longer tridiagonal. The explicit method is now implemented at r705 (see galdef parameter solution_method and description of popel.cc). Alternative methods using the full operator are possible, or more sophisticated operator splitting methods (see Press 1992), and these could also be investigated.

The diagnostics for a sample CSS run for electrons for the benchmark case are shown below. They show how the differencing-based timescale increases to a machine-accuracy solution at the end.

```
Network iteration 1 species 0 primary\_electrons (Z,A) = (-1,0)
propel: Entry
```

```
propel: Generating alpha for 2D

propel_diagnostics: call #1 dt=1000 total steps=1000 total time=1e+06 alpha timescale min max =85396.2  2.22185e+10 max frac diff wrt last CR=3.64742e+27
propel_diagnostics: call #2 dt=1000 total steps=2000 total time=2e+06 alpha timescale min max =171027  5.79181e+11 max frac diff wrt last CR=0.999652 gives min timescale=1.00035e+06 yrs
propel_diagnostics: call #3 dt=1000 total steps=3000 total time=3e+06 alpha timescale min max =174871  8.13567e+11 max frac diff wrt last CR=0.991165 gives min timescale=1.00891e+06 yrs
propel_diagnostics: call #4 dt=1000 total steps=4000 total time=4e+06 alpha timescale min max =174871  3.52647e+12 max frac diff wrt last CR=0.958903 gives min timescale=1.04286e+06 yrs
propel_diagnostics: call #5 dt=1000 total steps=5000 total time=5e+06 alpha timescale min max =174871  2.1875e+12 max frac diff wrt last CR=0.892462 gives min timescale=1.1205e+06 yrs
propel_diagnostics: call #6 dt=1000 total steps=6000 total time=6e+06 alpha timescale min max =174871  8.94e+13 max frac diff wrt last CR=0.804674 gives min timescale=1.24274e+06 yrs
propel_diagnostics: call #7 dt=1000 total steps=7000 total time=7e+06 alpha timescale min max =174871  1.60959e+13 max frac diff wrt last CR=0.71275 gives min timescale=1.40302e+06 yrs
propel_diagnostics: call #8 dt=1000 total steps=8000 total time=8e+06 alpha timescale min max =174871  1.52875e+13 max frac diff wrt last CR=0.626537 gives min timescale=1.59608e+06 yrs
propel_diagnostics: call #9 dt=1000 total steps=9000 total time=9e+06 alpha timescale min max =174871  8.36239e+12 max frac diff wrt last CR=0.549826 gives min timescale=1.81876e+06 yrs
propel_diagnostics: call #10 dt=1000 total steps=10000 total time=1e+07 alpha timescale min max =174871  1.62163e+13 max frac diff wrt last CR=0.483571 gives min timescale=2.06795e+06 yrs
propel_diagnostics: call #11 dt=1000 total steps=11000 total time=1.1e+07 alpha timescale min max =174871  3.33579e+14 max frac diff wrt last CR=0.42714 gives min timescale=2.34115e+06 yrs
propel_diagnostics: call #12 dt=1000 total steps=12000 total time=1.2e+07 alpha timescale min max =174871  1.8009e+13 max frac diff wrt last CR=0.378773 gives min timescale=2.64011e+06 yrs
propel_diagnostics: call #13 dt=1000 total steps=13000 total time=1.3e+07 alpha timescale min max =174871  1.86568e+13 max frac diff wrt last CR=0.337276 gives min timescale=2.96493e+06 yrs
propel_diagnostics: call #14 dt=1000 total steps=14000 total time=1.4e+07 alpha timescale min max =174871  6.36136e+14 max frac diff wrt last CR=0.301574 gives min timescale=3.31593e+06 yrs
propel_diagnostics: call #15 dt=1000 total steps=15000 total time=1.5e+07 alpha timescale min max =174871  3.55402e+14 max frac diff wrt last CR=0.270741 gives min timescale=3.69357e+06 yrs
propel_diagnostics: call #16 dt=1000 total steps=16000 total time=1.6e+07 alpha timescale min max =174871  1.81765e+14 max frac diff wrt last CR=0.243999 gives min timescale=4.09838e+06 yrs
propel_diagnostics: call #17 dt=1000 total steps=17000 total time=1.7e+07 alpha timescale min max =174871  2.57299e+14 max frac diff wrt last CR=0.220705 gives min timescale=4.53093e+06 yrs
propel_diagnostics: call #18 dt=1000 total steps=18000 total time=1.8e+07 alpha timescale min max =174871  2.94305e+14 max frac diff wrt last CR=0.2005 gives min timescale=4.98754e+06 yrs
propel_diagnostics: call #19 dt=1000 total steps=19000 total time=1.9e+07 alpha timescale min max =174871  3.0842e+14 max frac diff wrt last CR=0.182954 gives min timescale=5.46587e+06 yrs
propel_diagnostics: call #20 dt=1000 total steps=20000 total time=2e+07 alpha timescale min max =174871  3.13422e+14 max frac diff wrt last CR=0.167452 gives min timescale=5.97184e+06 yrs
propel_diagnostics: call #21 dt=1000 total steps=21000 total time=2.1e+07 alpha timescale min max =174871  3.15216e+14 max frac diff wrt last CR=0.153703 gives min timescale=6.50604e+06 yrs
propel_diagnostics: call #22 dt=1000 total steps=22000 total time=2.2e+07 alpha timescale min max =174871  3.15879e+14 max frac diff wrt last CR=0.141462 gives min timescale=7.06903e+06 yrs
propel_diagnostics: call #23 dt=1000 total steps=23000 total time=2.3e+07 alpha timescale min max =174871  3.86581e+14 max frac diff wrt last CR=0.130524 gives min timescale=7.6614e+06 yrs
propel_diagnostics: call #24 dt=1000 total steps=24000 total time=2.4e+07 alpha timescale min max =174871  3.16209e+14 max frac diff wrt last CR=0.120719 gives min timescale=8.28373e+06 yrs
propel_diagnostics: call #25 dt=1000 total steps=25000 total time=2.5e+07 alpha timescale min max =174871  3.1629e+14 max frac diff wrt last CR=0.111899 gives min timescale=8.9366e+06 yrs
propel_diagnostics: call #26 dt=1000 total steps=26000 total time=2.6e+07 alpha timescale min max =174871  3.16298e+14 max frac diff wrt last CR=0.103943 gives min timescale=9.62062e+06 yrs
propel_diagnostics: call #27 dt=1000 total steps=27000 total time=2.7e+07 alpha timescale min max =174871  3.16325e+14 max frac diff wrt last CR=0.0967457 gives min timescale=1.03364e+07 yrs
propel_diagnostics: call #28 dt=1000 total steps=28000 total time=2.8e+07 alpha timescale min max =174871  3.16312e+14 max frac diff wrt last CR=0.0902418 gives min timescale=1.10813e+07 yrs
propel_diagnostics: call #29 dt=1000 total steps=29000 total time=2.9e+07 alpha timescale min max =174871  3.16313e+14 max frac diff wrt last CR=0.0844667 gives min timescale=1.1839e+07 yrs
propel_diagnostics: call #30 dt=1000 total steps=30000 total time=3e+07 alpha timescale min max =174871  3.1631e+14 max frac diff wrt last CR=0.079193 gives min timescale=1.26274e+07 yrs
propel_diagnostics: call #31 dt=1000 total steps=31000 total time=3.1e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0743655 gives min timescale=1.34471e+07 yrs
propel_diagnostics: call #32 dt=1000 total steps=32000 total time=3.2e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0699363 gives min timescale=1.42987e+07 yrs
propel_diagnostics: call #33 dt=1000 total steps=33000 total time=3.3e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0658634 gives min timescale=1.51829e+07 yrs
propel_diagnostics: call #34 dt=1000 total steps=34000 total time=3.4e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0621104 gives min timescale=1.61004e+07 yrs
propel_diagnostics: call #35 dt=1000 total steps=35000 total time=3.5e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0586453 gives min timescale=1.70517e+07 yrs
propel_diagnostics: call #36 dt=1000 total steps=36000 total time=3.6e+07 alpha timescale min max =174871  1.08361e+15 max frac diff wrt last CR=0.0554399 gives min timescale=1.80375e+07 yrs
propel_diagnostics: call #37 dt=1000 total steps=37000 total time=3.7e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0524694 gives min timescale=1.90587e+07 yrs
propel_diagnostics: call #38 dt=1000 total steps=38000 total time=3.8e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0497118 gives min timescale=2.0116e+07 yrs
propel_diagnostics: call #39 dt=1000 total steps=39000 total time=3.9e+07 alpha timescale min max =174871  3.16294e+14 max frac diff wrt last CR=0.0471475 gives min timescale=2.121e+07 yrs
propel_diagnostics: call #40 dt=1000 total steps=40000 total time=4e+07 alpha timescale min max =174871  4.58477e+14 max frac diff wrt last CR=0.0447592 gives min timescale=2.23418e+07 yrs
propel_diagnostics: call #41 dt=1000 total steps=41000 total time=4.1e+07 alpha timescale min max =174871  1.24799e+15 max frac diff wrt last CR=0.0425316 gives min timescale=2.3512e+07 yrs
......
propel_diagnostics: call #406 dt=1000 total steps=406000 total time=4.06e+08 alpha timescale min max =174871  5.69779e+15 max frac diff wrt last CR=2.89966e-07 gives min timescale=3.44868e+12 yrs
propel_diagnostics: call #407 dt=1000 total steps=407000 total time=4.07e+08 alpha timescale min max =174871  1.90062e+17 max frac diff wrt last CR=2.80662e-07 gives min timescale=3.56301e+12 yrs
propel_diagnostics: call #408 dt=1000 total steps=408000 total time=4.08e+08 alpha timescale min max =174871  3.94735e+15 max frac diff wrt last CR=2.71656e-07 gives min timescale=3.68112e+12 yrs
propel_diagnostics: call #409 dt=1000 total steps=409000 total time=4.09e+08 alpha timescale min max =174871  1.01301e+17 max frac diff wrt last CR=2.62939e-07 gives min timescale=3.80316e+12 yrs
propel_diagnostics: call #410 dt=1000 total steps=410000 total time=4.1e+08 alpha timescale min max =174871  1.99103e+16 max frac diff wrt last CR=2.54502e-07 gives min timescale=3.92925e+12 yrs
.....

propel_diagnostics: call #836 dt=1000 total steps=836000 total time=8.36e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=1.24095e-13 gives min timescale=8.05833e+18 yrs
propel_diagnostics: call #837 dt=1000 total steps=837000 total time=8.37e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=1.24012e-13 gives min timescale=8.06373e+18 yrs
propel_diagnostics: call #838 dt=1000 total steps=838000 total time=8.38e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=8.60299e-14 gives min timescale=1.16239e+19 yrs
propel_diagnostics: call #839 dt=1000 total steps=839000 total time=8.39e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=9.98029e-14 gives min timescale=1.00198e+19 yrs
propel_diagnostics: call #840 dt=1000 total steps=840000 total time=8.4e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=1.53419e-14 gives min timescale=6.5181e+19 yrs
propel_diagnostics: call #841 dt=1000 total steps=841000 total time=8.41e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=0 gives min timescale=inf yrs
propel_diagnostics: call #842 dt=1000 total steps=842000 total time=8.42e+08 alpha timescale min max =174871  5.67184e+16 max frac diff wrt last CR=0 gives min timescale=inf yrs
```

Here is also a run for protons with no reacceleration and Dxx=constant (i.e. the simplest case, since no energy losses either). It reaches a solution with the machine accuracy after 8.6 10^8 years.

```
 Network iteration 1 species 0 Hydrogen_1 (Z,A) = (1,1)
propel: Entry
propel: Generating alpha for 2D

propel_diagnostics: call #1 dt=1000 total steps=1000 total time=1e+06 alpha timescale min max =98821.4  1.72529e+06 max frac diff wrt last CR=9.16648e+20
propel_diagnostics: call #2 dt=1000 total steps=2000 total time=2e+06 alpha timescale min max =171620  3.95304e+06 max frac diff wrt last CR=0.999593 gives min timescale=1.00041e+06 yrs
propel_diagnostics: call #3 dt=1000 total steps=3000 total time=3e+06 alpha timescale min max =289024  6.44136e+06 max frac diff wrt last CR=0.989835 gives min timescale=1.01027e+06 yrs
propel_diagnostics: call #4 dt=1000 total steps=4000 total time=4e+06 alpha timescale min max =444393  9.07956e+06 max frac diff wrt last CR=0.942108 gives min timescale=1.06145e+06 yrs
propel_diagnostics: call #5 dt=1000 total steps=5000 total time=5e+06 alpha timescale min max =630192  1.18098e+07 max frac diff wrt last CR=0.854741 gives min timescale=1.16995e+06 yrs
propel_diagnostics: call #6 dt=1000 total steps=6000 total time=6e+06 alpha timescale min max =843964  1.46033e+07 max frac diff wrt last CR=0.753331 gives min timescale=1.32744e+06 yrs
propel_diagnostics: call #7 dt=1000 total steps=7000 total time=7e+06 alpha timescale min max =1.08481e+06  1.74459e+07 max frac diff wrt last CR=0.655876 gives min timescale=1.52468e+06 yrs
propel_diagnostics: call #8 dt=1000 total steps=8000 total time=8e+06 alpha timescale min max =1.3525e+06  2.03305e+07 max frac diff wrt last CR=0.569451 gives min timescale=1.75608e+06 yrs
propel_diagnostics: call #9 dt=1000 total steps=9000 total time=9e+06 alpha timescale min max =1.64709e+06  2.32544e+07 max frac diff wrt last CR=0.495392 gives min timescale=2.0186e+06 yrs
propel_diagnostics: call #10 dt=1000 total steps=10000 total time=1e+07 alpha timescale min max =1.96884e+06  2.62171e+07 max frac diff wrt last CR=0.432776 gives min timescale=2.31066e+06 yrs
propel_diagnostics: call #11 dt=1000 total steps=11000 total time=1.1e+07 alpha timescale min max =2.31804e+06  2.02581e+08 max frac diff wrt last CR=0.380016 gives min timescale=2.63147e+06 yrs
propel_diagnostics: call #12 dt=1000 total steps=12000 total time=1.2e+07 alpha timescale min max =2.69508e+06  1.13506e+11 max frac diff wrt last CR=0.335494 gives min timescale=2.98068e+06 yrs
propel_diagnostics: call #13 dt=1000 total steps=13000 total time=1.3e+07 alpha timescale min max =3.10031e+06  6.61594e+09 max frac diff wrt last CR=0.297779 gives min timescale=3.3582e+06 yrs
propel_diagnostics: call #14 dt=1000 total steps=14000 total time=1.4e+07 alpha timescale min max =3.5341e+06  2.24319e+09 max frac diff wrt last CR=0.265669 gives min timescale=3.76409e+06 yrs
propel_diagnostics: call #15 dt=1000 total steps=15000 total time=1.5e+07 alpha timescale min max =3.9968e+06  8.42828e+08 max frac diff wrt last CR=0.238181 gives min timescale=4.19849e+06 yrs
propel_diagnostics: call #16 dt=1000 total steps=16000 total time=1.6e+07 alpha timescale min max =4.48876e+06  5.7233e+09 max frac diff wrt last CR=0.214519 gives min timescale=4.66158e+06 yrs
propel_diagnostics: call #17 dt=1000 total steps=17000 total time=1.7e+07 alpha timescale min max =5.0103e+06  8.91699e+08 max frac diff wrt last CR=0.19404 gives min timescale=5.15359e+06 yrs
propel_diagnostics: call #18 dt=1000 total steps=18000 total time=1.8e+07 alpha timescale min max =5.56173e+06  1.8658e+09 max frac diff wrt last CR=0.17622 gives min timescale=5.67472e+06 yrs

...........

propel_diagnostics: call #101 dt=1000 total steps=101000 total time=1.01e+08 alpha timescale min max =5.55127e+06  1.55213e+11 max frac diff wrt last CR=0.00408963 gives min timescale=2.44521e+08
propel_diagnostics: call #102 dt=1000 total steps=102000 total time=1.02e+08 alpha timescale min max =5.54996e+06  1.30176e+11 max frac diff wrt last CR=0.00396359 gives min timescale=2.52297e+08
propel_diagnostics: call #103 dt=1000 total steps=103000 total time=1.03e+08 alpha timescale min max =5.5487e+06  1.11512e+11 max frac diff wrt last CR=0.00384168 gives min timescale=2.60303e+08 y
propel_diagnostics: call #104 dt=1000 total steps=104000 total time=1.04e+08 alpha timescale min max =5.54749e+06  7.69655e+11 max frac diff wrt last CR=0.00372377 gives min timescale=2.68545e+08
propel_diagnostics: call #105 dt=1000 total steps=105000 total time=1.05e+08 alpha timescale min max =5.54632e+06  9.78229e+10 max frac diff wrt last CR=0.0036097 gives min timescale=2.77032e+08 y


...........

ropel_diagnostics: call #190 dt=1000 total steps=190000 total time=1.9e+08 alpha timescale min max =5.51754e+06  2.93882e+10 max frac diff wrt last CR=0.000276813 gives min timescale=3.61255e+09 y
propel_diagnostics: call #191 dt=1000 total steps=191000 total time=1.91e+08 alpha timescale min max =5.51749e+06  3.15509e+10 max frac diff wrt last CR=0.000268587 gives min timescale=3.72319e+09
propel_diagnostics: call #192 dt=1000 total steps=192000 total time=1.92e+08 alpha timescale min max =5.51744e+06  3.39667e+10 max frac diff wrt last CR=0.000260604 gives min timescale=3.83724e+09
```

```
propel_diagnostics: call #193 dt=1000 total steps=193000 total time=1.93e+08 alpha timescale min max =5.5174e+06  3.6681e+10 max frac diff wrt last CR=0.000252856 gives min timescale=3.95482e+09 y
propel_diagnostics: call #194 dt=1000 total steps=194000 total time=1.94e+08 alpha timescale min max =5.51736e+06  3.97509e+10 max frac diff wrt last CR=0.000245337 gives min timescale=4.07603e+09
propel_diagnostics: call #195 dt=1000 total steps=195000 total time=1.95e+08 alpha timescale min max =5.51732e+06  4.3249e+10 max frac diff wrt last CR=0.000238039 gives min timescale=4.20099e+09
propel_diagnostics: call #196 dt=1000 total steps=196000 total time=1.96e+08 alpha timescale min max =5.51728e+06  4.7269e+10 max frac diff wrt last CR=0.000230957 gives min timescale=4.32981e+09

.....
 call #225 dt=1000 total steps=225000 total time=2.25e+08 alpha timescale min max =5.51658e+06  2.41958e+13 max frac diff wrt last CR=9.58694e-05 gives min timescale=1.04309e+10 yrs
 call #226 dt=1000 total steps=226000 total time=2.26e+08 alpha timescale min max =5.51656e+06  2.76663e+13 max frac diff wrt last CR=9.29956e-05 gives min timescale=1.07532e+10 yrs
 call #227 dt=1000 total steps=227000 total time=2.27e+08 alpha timescale min max =5.51655e+06  4.12834e+13 max frac diff wrt last CR=9.02072e-05 gives min timescale=1.10856e+10 yrs
 call #228 dt=1000 total steps=228000 total time=2.28e+08 alpha timescale min max =5.51654e+06  3.92432e+12 max frac diff wrt last CR=8.75017e-05 gives min timescale=1.14283e+10 yrs
 call #229 dt=1000 total steps=229000 total time=2.29e+08 alpha timescale min max =5.51653e+06  2.93033e+12 max frac diff wrt last CR=8.48767e-05 gives min timescale=1.17818e+10 yrs
....

 call #854 dt=1000 total steps=854000 total time=8.54e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=1.37263e-13 gives min timescale=7.28527e+18 yrs
 call #855 dt=1000 total steps=855000 total time=8.55e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=1.96174e-13 gives min timescale=5.09752e+18 yrs
 call #856 dt=1000 total steps=856000 total time=8.56e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=1.77353e-13 gives min timescale=5.63846e+18 yrs
 call #857 dt=1000 total steps=857000 total time=8.57e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=1.75122e-13 gives min timescale=5.7103e+18 yrs
 call #858 dt=1000 total steps=858000 total time=8.58e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=1.06768e-13 gives min timescale=9.3661e+18 yrs
 call #859 dt=1000 total steps=859000 total time=8.59e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
 call #860 dt=1000 total steps=860000 total time=8.6e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
 call #861 dt=1000 total steps=861000 total time=8.61e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
 call #862 dt=1000 total steps=862000 total time=8.62e+08 alpha timescale min max =5.51563e+06  3.67754e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
```

and for protons with reacceleration and D_{xx} with $\delta = 0.33$:

```
propel_diagnostics: call #1 dt=1000 total steps=1000 total time=1e+06 alpha timescale min max =85585.3  2.19767e+07 max frac diff wrt last CR=9.4315e+20
propel_diagnostics: call #2 dt=1000 total steps=2000 total time=2e+06 alpha timescale min max =171257  1.88135e+08 max frac diff wrt last CR=0.999648 gives min timescale=1.00035e+06 yrs
propel_diagnostics: call #3 dt=1000 total steps=3000 total time=3e+06 alpha timescale min max =190546  1.22079e+08 max frac diff wrt last CR=0.990962 gives min timescale=1.00912e+06 yrs
propel_diagnostics: call #4 dt=1000 total steps=4000 total time=4e+06 alpha timescale min max =183844  1.36473e+09 max frac diff wrt last CR=0.955672 gives min timescale=1.04638e+06 yrs
propel_diagnostics: call #5 dt=1000 total steps=5000 total time=5e+06 alpha timescale min max =180703  4.30048e+09 max frac diff wrt last CR=0.88377 gives min timescale=1.13152e+06 yrs
propel_diagnostics: call #6 dt=1000 total steps=6000 total time=6e+06 alpha timescale min max =178969  1.14657e+10 max frac diff wrt last CR=0.790886 gives min timescale=1.26441e+06 yrs
propel_diagnostics: call #7 dt=1000 total steps=7000 total time=7e+06 alpha timescale min max =177897  3.55195e+09 max frac diff wrt last CR=0.695342 gives min timescale=1.43814e+06 yrs
propel_diagnostics: call #8 dt=1000 total steps=8000 total time=8e+06 alpha timescale min max =177181  1.48549e+10 max frac diff wrt last CR=0.606934 gives min timescale=1.64762e+06 yrs
propel_diagnostics: call #9 dt=1000 total steps=9000 total time=9e+06 alpha timescale min max =176681  3.81669e+09 max frac diff wrt last CR=0.52909 gives min timescale=1.89004e+06 yrs
propel_diagnostics: call #10 dt=1000 total steps=10000 total time=1e+07 alpha timescale min max =176319  2.25285e+09 max frac diff wrt last CR=0.462085 gives min timescale=2.1641e+06 yrs
propel_diagnostics: call #11 dt=1000 total steps=11000 total time=1.1e+07 alpha timescale min max =176050  1.85755e+09 max frac diff wrt last CR=0.405324 gives min timescale=2.46716e+06 yrs
propel_diagnostics: call #12 dt=1000 total steps=12000 total time=1.2e+07 alpha timescale min max =175846  2.98466e+10 max frac diff wrt last CR=0.357053 gives min timescale=2.80071e+06 yrs
propel_diagnostics: call #13 dt=1000 total steps=13000 total time=1.3e+07 alpha timescale min max =175689  2.78658e+10 max frac diff wrt last CR=0.315918 gives min timescale=3.16538e+06 yrs

.......
propel_diagnostics: call #214 dt=1000 total steps=214000 total time=2.14e+08 alpha timescale min max =175048  1.71558e+18 max frac diff wrt last CR=6.74427e-05 gives min timescale=1.48274e+10 yrs
propel_diagnostics: call #215 dt=1000 total steps=215000 total time=2.15e+08 alpha timescale min max =175048  2.23556e+17 max frac diff wrt last CR=6.50327e-05 gives min timescale=1.53769e+10 yrs
propel_diagnostics: call #216 dt=1000 total steps=216000 total time=2.16e+08 alpha timescale min max =175048  1.2513e+17 max frac diff wrt last CR=6.27068e-05 gives min timescale=1.59472e+10 yrs
propel_diagnostics: call #217 dt=1000 total steps=217000 total time=2.17e+08 alpha timescale min max =175048  2.09001e+18 max frac diff wrt last CR=6.04622e-05 gives min timescale=1.65393e+10 yrs
propel_diagnostics: call #218 dt=1000 total steps=218000 total time=2.18e+08 alpha timescale min max =175048  3.26439e+17 max frac diff wrt last CR=5.82961e-05 gives min timescale=1.71538e+10 yrs
propel_diagnostics: call #219 dt=1000 total steps=219000 total time=2.19e+08 alpha timescale min max =175048  1.01979e+20 max frac diff wrt last CR=5.62059e-05 gives min timescale=1.77917e+10 yrs
.......
propel_diagnostics: call #712 dt=1000 total steps=712000 total time=7.12e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=2.00948e-13 gives min timescale=4.97641e+18 yrs
propel_diagnostics: call #713 dt=1000 total steps=713000 total time=7.13e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=1.64118e-13 gives min timescale=6.09316e+18 yrs
propel_diagnostics: call #714 dt=1000 total steps=714000 total time=7.14e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=1.10842e-13 gives min timescale=9.02186e+18 yrs
propel_diagnostics: call #715 dt=1000 total steps=715000 total time=7.15e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=4.07829e-14 gives min timescale=2.45201e+19 yrs
propel_diagnostics: call #716 dt=1000 total steps=716000 total time=7.16e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
propel_diagnostics: call #717 dt=1000 total steps=717000 total time=7.17e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
propel_diagnostics: call #718 dt=1000 total steps=718000 total time=7.18e+08 alpha timescale min max =175048  3.7852e+17 max frac diff wrt last CR=0 gives min timescale=inf yrs
```

Timestep modes 2 follows timestep mode 1 seamlessly as the following example shows. It uses with $10^9 - 10^3$ years with 100 steps per timestep in mode 1, and then continues in mode 2. The diagnostics show that the changeover is smooth, while as expected the α-based timescale is still too small in mode 1, but then starts to increase in steadily mode 2 (and eventually reaches 10^{18} years as in the mode-2-only example). Hence mode 1 can be used to accelerate the initial solution and mode 2 to refine this to an accurate solution, with an overall reduction in CPU requirements.

```
propel_diagnostics: call #6196 dt=1226 total steps=6196 total time=5e+11 alpha timescale min max =149011  3.99118e+12 max frac diff wrt last CR=2.86072e-05 gives min timescale=4.28562e+07 yrs conv
propel_diagnostics: call #6197 dt=1226 total steps=6197 total time=5e+11 alpha timescale min max =149015  4.30621e+12 max frac diff wrt last CR=2.83603e-05 gives min timescale=4.32293e+07 yrs conv
propel_diagnostics: call #6198 dt=1226 total steps=6198 total time=5e+11 alpha timescale min max =149019  4.70623e+12 max frac diff wrt last CR=2.81189e-05 gives min timescale=4.36005e+07 yrs conv
propel_diagnostics: call #6199 dt=1226 total steps=6199 total time=5e+11 alpha timescale min max =149022  5.22521e+12 max frac diff wrt last CR=2.78826e-05 gives min timescale=4.397e+07 yrs conver
propel_diagnostics: call #6200 dt=1226 total steps=6200 total time=5e+11 alpha timescale min max =149026  5.91887e+12 max frac diff wrt last CR=2.76513e-05 gives min timescale=4.43377e+07 yrs conv
propel:     timestep_mode=2
propel: timestep_mode 2: using  method 2 fully time explicit,  turbo=1 timestep=1226 yr
propel_diagnostics: call #6201 dt=1226 total steps=6201 total time=5e+11 alpha timescale min max =203038  3.63635e+12 max frac diff wrt last CR=0.0127593 gives min timescale=96086.2 yrs converged=
propel_diagnostics: call #6202 dt=1226 total steps=6202 total time=5e+11 alpha timescale min max =267739  1.83048e+12 max frac diff wrt last CR=0.00846563 gives min timescale=144820 yrs converged=
propel_diagnostics: call #6203 dt=1226 total steps=6203 total time=5e+11 alpha timescale min max =342633  8.45072e+12 max frac diff wrt last CR=0.00598687 gives min timescale=204781 yrs converged=
propel_diagnostics: call #6204 dt=1226 total steps=6204 total time=5e+11 alpha timescale min max =426933  7.15266e+12 max frac diff wrt last CR=0.00446404 gives min timescale=274639 yrs converged=
propel_diagnostics: call #6205 dt=1226 total steps=6205 total time=5e+11 alpha timescale min max =519745  1.80703e+12 max frac diff wrt last CR=0.00347308 gives min timescale=353000 yrs converged=
propel_diagnostics: call #6206 dt=1226 total steps=6206 total time=5e+11 alpha timescale min max =620199  1.79934e+12 max frac diff wrt last CR=0.00279472 gives min timescale=438683 yrs converged=
propel_diagnostics: call #6207 dt=1226 total steps=6207 total time=5e+11 alpha timescale min max =727512  1.79172e+12 max frac diff wrt last CR=0.00230985 gives min timescale=530769 yrs converged=
propel_diagnostics: call #6208 dt=1226 total steps=6208 total time=5e+11 alpha timescale min max =841018  1.91966e+12 max frac diff wrt last CR=0.00195048 gives min timescale=628560 yrs converged=
propel_diagnostics: call #6209 dt=1226 total steps=6209 total time=5e+11 alpha timescale min max =960159  1.28067e+13 max frac diff wrt last CR=0.00167597 gives min timescale=731515 yrs converged=
propel_diagnostics: call #6210 dt=1226 total steps=6210 total time=5e+11 alpha timescale min max =1.08447e+06  1.76921e+12 max frac diff wrt last CR=0.00146091 gives min timescale=839201 yrs conve
propel_diagnostics: call #6211 dt=1226 total steps=6211 total time=5e+11 alpha timescale min max =1.21357e+06  1.76183e+12 max frac diff wrt last CR=0.00128881 gives min timescale=951262 yrs conve
propel_diagnostics: call #6212 dt=1226 total steps=6212 total time=5e+11 alpha timescale min max =1.34713e+06  1.75452e+12 max frac diff wrt last CR=0.00114859 gives min timescale=1.06739e+06 yrs
......
```

2. The accelerated solution: start timestep 10^9 yr, end timestep 1 yr, ratio 0.25. The suggested repeat per timestep of 20 (section 5.5.3) leads to deviations of up to 20% from the accurate solution, although at most energies and positions the deviations are within 10%. The largest deviations are at large R and large z. Increasing to the number of repeats to 100, 1000, 10000 improves the accuracy, with the latter giving deviations from the accurate solution of 0.1% and hence verifying the validity of such solutions. NB The use of the propel_diagnostics.cc diagnostics for the accelerated solution will need to be clarified since the variable timestep may affect the interpretation; the only reliable check is to compare with a CSS solution.

The **conclusions** of these studies so far are: (i) the accelerated solutions should use at least 100 repeats per timestep, and preferably 1000 or more, (ii) the solutions should be compared for various timesteps and the accelerated solution compared with a long CSS run, (iii) the diagnostics provided by propel_diagnostics should be activated (via the appropriate GALDEF parameters) and their output studied. The reference should ideally be a run for which the differencing-based timescale from propel_diagnostics becomes machine infinity everywhere in space and energy. The tests should be done for every species considered since e.g. protons, radioactive nuclei and electrons have quite different propagation characteristics.

(19 Jan 2011:) The *explicit time* method is now available in GALPROP, and tests with diagnostics will be added soon in this section.

12 Document history

This documents the history of this manual.

01 July 2009 AWS: new for v54 based on v50 manual

05 July 2009 AWS: added section on program architecture

23 July 2009 AWS: documented all new parameters

05 May 2010 AWS: description of format of galdef file

07 May 2010 AWS: description of CR electron source parameters

11 May 2010 AWS: description of nuclei files and headers

14 Sept 2010 AWS: convection differencing scheme etc explained

01 Oct 2010 AWS: explained ELENORM and NUCNORM in 3D nuclei file.

05 Oct 2010 AWS: added description of parameters source_norm and electron_source_norm provided by Gulli.

11 Oct 2010 AWS: updated descripion of B field model to reflect new and obsolete parameters.

21 Oct 2010 AWS: added warning about parameter use_symmetry

03 Nov 2010 AWS: started section on synchrotron calculation details

10 Nov 2010 AWS: started section on coordinate system; reference to Vladimirov etal 2010, and copy of enhancements from there.

16,24 Nov 2010 AWS: galdef parameter synchrotron: clarified use for energy losses

15-23 Dec 2010 AWS: added recommendation to do a test run with small timesteps. Added description of new parameters proton_norm_type, electron_norm_type, new value for inj_spectrum_type=dirac. More on timesteps.

31 Dec 2010 AWS: started section on tests.

13-17 Jan 2011 AWS 5.5.3: covered 3D as well as 2D case. Explained that SM98 described the wrong method. Formulated the real CN method. Formulated the explicit method. Updated authors affiliations, added Andrey.

19-21 Jan 2011 AWS explicit method implementation described, and corresponding parameter explained. Convergence parameter described.

25 Jan 2011 AWS added description of parameters network_iter_compl and network_iter_sec

03 Feb 2011 AWS added example of changeover from timestep mode 1 to 2. Described turbo method in galdef parameters.

09 Mar 2011 AWS description of Stokes calculation, calculation of synchrotron emissivities.

06 May 2011 AWS started analytical solution for decay case

20 May 2011 AWS B_field_parameters: no restriction on number of parameters

10 Jun 2011 AWS synchrotron emissivity formulae corrected

05 Jul 2011 AWS free-free absorption parameters added

07 Sep 2011 AWS free-free maps always output. in output description, skymap names for healpix described.

19 Sep 2011 AWS synchrotron polarization angle maps described

22 Sep 2011 AWS copied from galprop_v54.tex and changed title accordingly. Described synch. polarized fraction.

16 Nov 2011 AWS copied from galprop_55.tex, changed title, no longer with version number.

16 Dec 2011 AWS updated notes on synchrotron options.

07 Feb 2012 AWS corrected formula for exponential B-field: R_o was missing.

20 Feb 2012 AWS added synchrotron emissivity units and Q, U emissivity files.

18 Apr 2012 AWS documented Huang option for pi0_decay and c++ option for bremss.

20 Mar 2013 AWS documented new convection model with wind starting at given z.
11 Apr 2013 AWS Healpix skymap intensity units stated: no Eg^2 factor

References

Badhwar, G. D., Stephens, S. A., & Golden, R. L. 1977, Phys. Rev. D, 15, 820

Berezinskii, V. S., Bulanov, S. V., Dogiel, V. A., Ginzburg, V. L., & Ptuskin, V. S. 1990, Astrophysics of Cosmic Rays (Amsterdam: North Holland)

Broadbent, A., et al. 1990, Proc. 21st Int. Cosmic Ray Conf., 3, 229

Bronfman, L., Cohen, R. S., Alvarez, H., May, J., & Thaddeus, P. 1988, ApJ, 324, 248

Cordes, J. M., Weisberg, J. M., Frail, D. A., Spangler, S. R., & Ryan, M. 1991, Nature, 354, 121

Cox, P., Krügel, E., & Mezger, P. G. 1986, A&A, 155, 380

Dermer, C. D. 1986a, ApJ, 307, 47

Dermer, C. D. 1986b, A&A, 157, 223

Dickey, J. M., & Lockman, F. J. 1990, Ann. Rev. Astron. Astrophys., 28, 215

Dwek, E., et al. 1997, ApJ, 475, 565

Ferrando, P., Webber, W. R., Goret, P., Kish, J. C., Schrier, D. A., Soutoul, A., & Testerd, O. 1988, Phys. Rev. C, 37, 1490

Freudenreich, H. T. 1998, ApJ, 492, 495

Gaisser, T. K., & Schaefer, R. K. 1992, ApJ, 394, 174

Gordon, M. A., & Burton, W. B. 1976, ApJ, 208, 346

Groom, D. E., et al. 2000, Europ. Phys. J., C15, 1

Heiles, C. 1996, in ASP Conf. Ser. 97, Polarimetry of the Interstellar Medium, ed. W. G. Roberge & D. C. B. Whittet (San Francisco: ASP), 457

Koch, H. W., & Motz, J. W. 1959, Rev. Mod. Phys., 31, 920

Lazio, T. J. W., & Cordes, J. M. 1998a, ApJ, 497, 238

Lazio, T. J. W., & Cordes, J. M. 1998b, ApJ, 505, 715

Letaw, J. R., Silberberg, R., & Tsao, C. H. 1983, ApJS, 51, 271

Meyer, J.-P., Drury, L. O'C., & Ellison, D. C. 1998, Space Sci. Rev., 86, 179

Moiseev, A. A., & Ormes, J. F. 1997, Astropart. Phys., 6, 379

Moskalenko, I. V., & Strong, A. W. 1998, ApJ, 493, 694

Moskalenko, I. V., & Strong, A. W. 1999, Phys. Rev. D, 60, #063003

Moskalenko, I. V., & Strong, A. W. 2000a, ApJ, 528, 357

Moskalenko, I. V., & Strong, A. W. 2000b, Astrophys. Space Sci., 272, 247

Moskalenko, I. V., Mashnik, S. G., & Strong, A. W. 2001a, Proc. 27th Int. Cosmic Ray Conf., OG 1.3, (astro-ph/0106502)

Moskalenko, I. V., Strong, A. W., & Reimer, O. 1998, A&A, 338, L75

Moskalenko, I. V., Strong, A. W., Ormes, J. F., & Potgieter, M. S. 2001b, ApJ, submitted (astro-ph/0106567)

Moskalenko, I. V., Porter, T. A., & Strong, A. W. 2006, ApJ, , , in press, (astro-ph/0511149)

Porter, T. A., & Strong, A. W. a, Proc. 29th Int. Cosmic Ray Conf., 2005 , (stro-ph/0507119)

Press, W. H., et al. 1992, Numerical Recipes in FORTRAN (2d ed.; Cambridge: Cambridge Univ. Press)

Roesler, S., Engel, R., & Ranft, J. 1998, Phys. Rev. D, 57, 2889

Seo, E. S., & Ptuskin, V. S. 1994, ApJ, 431, 705

Simon, M., Molnar, A., & Roesler, S. 1998, ApJ, 499, 250

Sodrowski, T. J., et al. 1997, ApJ, 480, 173

Stecker, F. W. 1970, Astrophys. Space Sci., 6, 377

Stephens, S. A., & Badhwar, G. D. 1981, Astrophys. Space Sci., 76, 213

Strong, A. W., & Mattox, J. R. 1996, A&A, 308, L21

Strong, A. W., & Moskalenko, I. V. 1998, ApJ, 509, 212

Strong, A. W., & Moskalenko, I. V. 1999, in Horizons in World Physics 230, Topics in Cosmic Ray Astrophysics, ed. M. A. DuVernois (New York: Nova Scientific), 81

Strong, A. W., & Moskalenko, I. V. (, Adv. Space Res., 27, 2001astro-ph/0101068)

Strong, A. W., Moskalenko, I. V., & Reimer, O. 2000, ApJ, 537, 763; Erratum: 2000, ApJ, 541, 1109

Strong, A. W., & Moskalenko, I. V. (, Proc. 27th Int. Cosmic Ray Conf., OG 1.3, 2001bastro-ph/0106504)

Strong, A. W., & Moskalenko, I. V. 2001c, Proc. 27th Int. Cosmic Ray Conf., OG 1.3, in press (astro-ph/0106505)

Strong, A. W., Moskalenko, I. V., & Reimer, O. 2004a, ApJ, 613, 956

Strong, A. W., Moskalenko, I. V., & Reimer, O. 2004b, ApJ, 613, 962

Strong, A. W., Moskalenko, I. V., Reimer, O., Digel, S., Diehl, R. 2004c, A&A, 422, L47

Strong, A. W., Moskalenko, I. V., & Ptuskin, V. S. 2007, Ann. Rev. Nucl. Part. Sci., 57, 285

Strong, A. W., Moskalenko, I. V., Porter, T. A., Jóhannesson, G., Orlando, E., Digel, S.W., 2009, Proc 31st ICRC, arxiv.org/abs/0907.0559

Tan, L. C., & Ng, L. K. 1983a, J. Phys. G: Nucl. Part. Phys., 9, 227

Tan, L. C., & Ng, L. K. 1983b, J. Phys. G: Nucl. Part. Phys., 9, 1289

Taylor, J. H., & Cordes, J. M. 1993, ApJ, 411, 674

Vallée, J. P. 1996, Fund. Cosmic Phys., 19, 1

Vladimirov, A. E., et al., submitted to Computer Physics Communications, arXiv:1008.3642

Wainscoat, R. J., et al. 1992, ApJS, 83, 111

Webber, W. R., Kish, J. C., & Schrier, D. A. 1990, Phys. Rev. C, 41, 566

Zirakashvili, V. N., Breitschwerdt, D., Ptuskin, V. S., & Völk, H. J. 1996, A&A, 311, 113