FINAL REPORT

Upgrade of the Semi-Deterministic Model to study the long term evolution of the space debris

ESA/ESOC Contract No. 15857/01/D/HK(SC)

ESA/ESOC Technical Supervisor: Rüdiger Jehn

A. Rossi, L. Anselmo, C. Pardini
ISTI–CNR, Pisa, Italy

G.B. Valsecchi
IASF–CNR, Roma, Italy

ISTITUTO DI SCIENZA E TECNOLOGIE
DELL'INFORMAZIONE "A. FAEDO"

Contents

1 The Space Debris Mitigation model, Version 3.0 (SDM 3.0)

The Space Debris Mitigation long-term analysis program (SDM) has been developed to study the long-term evolution of orbital debris and to evaluate the effectiveness of mitigation measures. The aim of the model is to follow, as much as possible, the actual orbital evolution of space objects. Therefore, for the large ones, each orbit is individually propagated. However, due to the very large number of small debris involved in the problem, some simplifying assumptions had to be introduced. In particular, it was not realistic to propagate all these particles individually; then, a sampling was introduced and only the representative particles were propagated.
The software has been developed for Unix/Linux machines and exploits the file system structure of this operative system. Apart from this, it is fully machine independent, since is does not use any library routine. All the code is written in Fortran 77. The region of space considered ranges from the altitude of 0 km (actually from the limit of the Earth's atmosphere, that can be chosen by the user as an input variable) to 40,000 km; this hollow sphere is divided in 800 altitude shells 50 km wide. The objects considered in SDM are divided in two categories. The objects present in space at the start of a simulation belong to the so called *historical population*; it is composed by all the objects with mass larger than 10^{-3} grams generated by past space activities and catastrophic events (e.g., launches, explosions, collisions, RORSAT's cooling fluid leaks, etc...). This population, obtained from any suitable existing model (e.g., the ESA's MASTER model) is processed by an external software which propagates the debris orbits for a given time span, once for all, and computes the debris densities (in objects/km^3), as a function of altitude and mass. These densities are stored as SDM working data files (see Sec.1.2). Then, the historical population is treated by SDM only in terms of densities and it is never propagated again during a run of the software. Moreover, the individual identity of any object is lost. This is an important feature of SDM, intended to save mass memory and CPU time. The second category of objects is the *running population*. It is composed by everything that is injected in space after the beginning of a simulation, that is satellites, rocket bodies and mission related objects injected with launches, fragments created by explosions and collisions, droplets produced by RORSAT-like events, solid rocket motor slag. The objects of the running population retain their identity, apart for sampling. If they are larger than a user defined threshold mass, their orbital state is propagated individually, otherwise a sampled object, representing a number of debris with the same mass (the sampling factor is again user defined), is instead propagated. At any computation step (the default value is one year), the collision probability is computed using the *background population* density, obtained by combining the historical and running population densities. During a simulation, the running population will grow according to some source mechanisms, such as explosions, collisions, leaks and launches, and will decrease according to the sink mechanisms, such as air drag and de-orbiting. The population, either in the form of densities or discrete objects, is subdivided in mass bins (the default number of mass bins is presently 10).
SDM was originally developed in the early 90's under an ESA Contract. Since SDM 1.0 was developed in 1993, several versions of the code have been released. This document describes

the main upgrades performed for SDM version 3.0. All the general features developed in the previous versions of SDM are not recalled here, unless necessary. They can be found in the Study Notes of the previous contracts [4], [5].

1.1 Software style

Though still written in Fortran 77, the software has been completely revised, paving the way for a Fortran 90 Version. In this respect, all the routines are now written with the

IMPLICIT none

style. Hence, each variable is explicitly declared, with its own type (e.g. integer, double precision, ...) and, therefore, tracked allowing a more rigorous control of the code.

1.2 The Pre-Processing software

The purpose of the SDM pre-processing software is to produce the input density files with the orbital evolution of the *historical population* for a given number of years, starting from a set of orbital elements given at a selected reference epoch. The generation of the input density files for SDM stems from a main population (e.g. the ESA's MASTER model) and, in case, one or more additional populations.
The pre-processing package is structured as follows: in the SDM package directories tree (Fig. 1), starting from the main directory SDM, there is the directory DENSHIS (where the historical density files are stored), then a subdirectory WORK containing executables, macros and initial conditions of the populations (e.g. the MASTER population orbital elements file), possibly stored in different subdirectories.

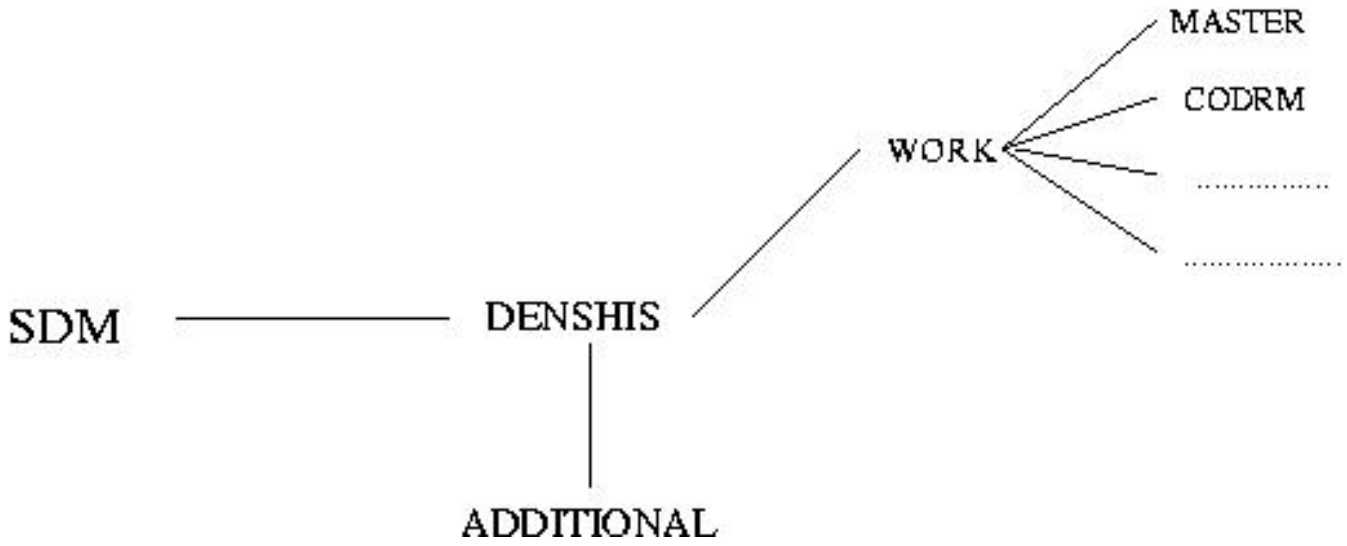

Figure 1: The directories tree of the pre-processing package.

All the Fortran files for the generation of the historical density files are located in the SDM/SOURCES

directory. A complete description of the input files for the pre-processing part of the software package is given later on in this section.
The main Fortran code is *creden.f* (A.3) and, along with the other subroutines, it is compiled by the makefile of SDM giving the command:

make creden

The executable files are then directly moved in the directory SDM/DENSHIS/WORK, devoted to the work related to the historical densities files. The executable file is *creden.x*, which creates the density of the given population.
The shell script, to automatically run all the executables and directly get the final density files in the appropriate directories, is called *creden* (B.1).
Running the macro *creden* (B.1), the user is asked to specify the path identifying the population file. Then the user is given two choices: if the population is the main one (e.g. the MASTER population) the densities are stored in the DENSHIS directory. Then the user has also the possibility to add another population to the main one (e.g. a new debris cloud from a recent fragmentation event, not included in the MASTER population file). The densities from this additional population can be stored in a directory called DENSHIS/ADDITIONAL. It is important to stress the fact that different contributions must refer (approximately) to the same epoch, since the program starts the propagation of the orbital elements assuming a common reference epoch.
Note that, unlike in the previous versions of SDM, the starting epoch of the propagation is not listed any more in the file *creden.inp*. The starting epoch is instead written at the beginning of the population file and is, therefore, the reference epoch of the population file.
To run *creden* (B.1) we need:

- one, or more, files with the elements of the objects: one is for the *standard* population (e.g. MASTER 2001 or CODRM-99). In principle, we can have as many input population files as we want. However, SDM will work with a maximum of one additional directory of density files (DENSHIS/ADDITIONAL) besides the *standard* one;
- the file *creden.inp* (C.7), containing the time span of the propagation and the flag for the choice of the orbital propagator (see later Sec. 2.1 and C.7);
- the file *densinp.dat* (C.6), with the information on the format of the data files read.

Note that in the input file with the orbital elements and the characteristics of the objects a set of default units is assumed. In particular the semimajor axis must be in km, the mass in kg, the diameter in m, the area in m^2 and all the angles in degrees. If any of this quantity is not expressed in the default units, it must be changed, *off line, before running the pre-processing.*
The density files have two lines of header. In the first line there are the legenda for the columns. In the second line, the lowest and highest limit of the altitude shells and the width of each shell (all in km), are listed.
Then, in each column, there is the density of objects in each altitude shell (in objects/km^3), for each mass bin. Therefore the typical file is 802 lines long, having two header lines and 800 altitude shells density lines, each of the latter composed by 10 columns (one for each mass bin).

In order to save disk space, the densities input files are written with the format: 1pd9.3. The same format applies also to the output density files produced after a run of SDM.
The historical density input files of the main population are then stored in the DENSHIS directory, while the density input files of the *additional* population are stored in the DENSHIS/ADDITIONAL directory.
In the SDM 3.0 version, *creden.f* (A.3) is also able to print as output the whole set of elements of the population, at a given time step (set by the user in the file *creden.inp* (C.7)). This allows the user to check the orbital distribution of the objects after the propagation and, mainly, to re-start the propagation in case further density files are needed, without re-running the creden software for the whole time span.
The initial population files and the related densities created with the pre-processing are stored also on a CD ROM so that a user may choose to run SDM without storing these files on the hard disk. See Sec. 2.6.4 for a detailed description of the procedures needed to run SDM from the CD ROM.

2 The new models inside SDM 3.0

The upgrade of SDM was mainly devoted to the addition of some new features and models for the sources and sinks mechanisms driving the long term space debris evolution. In this framework the following new models have been added:

- a new orbital propagator;
- a model for the production of solid rocket motor slag;
- a new breakup model for explosions and collisions;
- a new breakup model for low-velocity collisions;
- new, advanced mitigation options and scenarios.

In the following sections a detailed description of these new features will be given.

2.1 Orbital propagators

In SDM 2.0 all the orbital propagations were done with the Debris Cloud Propagator, a very fast and efficient orbital propagator including only the effect of the air drag. It allowed an extremely fast propagation of a reduced set of orbital elements (semimajor axis, eccentricity and inclination) [4].
The FOP orbital propagator has been added to SDM 3.0. FOP (Fast Orbit propagator) was originally developed as a stand alone code for the first SDM contract [4]. It is a very accurate propagator including all the relevant gravitational and non gravitational perturbations: geopotential harmonics, third body perturbations, solar radiation pressure (including shadows) and atmospheric drag. FOP uses the variation of parameters theory based on the integration of the

Lagrange planetary equations of the orbital elements for geopotential, luni-solar perturbations and solar radiation pressure. The adopted approach consisted of a separation of perturbations: the individual rates of change of the orbital elements due to geopotential, air-drag, luni-solar effects and solar radiation pressure are determined, and the results are superimposed in order to extrapolate the behavior of the orbit state with time.
FOP has never been used inside SDM since it is very demanding in terms of CPU time when applied to the huge number of objects treated by SDM during a 100-year run. Now the advancements in computing power since the early 90's make it marginally possible to use FOP within SDM. A flag, named *iprop* has been added to the input file *sdm.dat* (C.4). If *iprop* = 0 only the DCP (A.4) orbital propagator is used to propagate all the orbits during the run. If *iprop* = 1 only FOP is used as a propagator. If *iprop* = 2 both DCP and FOP are used according to the type of orbit to be propagated. Generally speaking, when *iprop = 2*, as a default, the circular Low Earth Orbits are propagated with DCP, while the other orbits where perturbations other then the atmospheric drag become relatively important, are propagated with FOP. In particular all the orbits with semimajor axis above the limit of validity of DCP specified by the parameter *rmxdcp* in the header file *datdcp.h* (A.8) are propagated by FOP. The limiting eccentricity discriminating between the orbits to be propagated with DCP and FOP is given by the parameter *limec* inside the header file *parpro.h* (A.15). Differently from DCP, FOP propagates the whole set of 6 orbital elements (a, e, i, Ω, ω and M). Therefore, within the code, the matrix *bin* containing the characteristics of an object has been enlarged to accommodate the whole set of elements. There is still the possibility to run SDM with a reduced set of 3 elements (a, e and i) and this is done by setting the parameter *npar_orb* either to 6 or to 3 inside the header file *parbin.h* (A.14). Accordingly all the input files have been changed, so that all the 6 orbital elements are listed and used, when applicable.
Preliminary tests indicate that a 100-year run of SDM 3.0 with FOP as the orbital propagator is several 10^4 times slower than the same run with DCP. At the moment it appears reasonable to use a few runs with FOP as "validators" for a whole campaign done with DCP, rather than as the *default* propagator. Nonetheless, in a near future the use of FOP as the *default* propagator appears a viable option with the advent of faster computers.

2.2 Solid rocket motor slag

A model for the production of slag and liner after a solid rocket motor burn has been added to SDM 3.0. The theoretical model is taken from MASTER 2001 [6] and the implementation is done independently.
Only the slag or liner particles are considered since the dust is below the SDM threshold mass (1 mg).
A diameter distribution of the particles is considered and later on transformed to mass (the main physical quantity used within SDM). The diameter distribution of the particles is given by:

$$N_{>d} = N^*_{>d^*} \left(\frac{d^*}{d} \right)^3 \tag{1}$$

where $N^*_{>d^*} = 1800$, $d^* = 5$ mm. These parameters were derived from radar measurements performed by MIT Lincoln Lab. for a suborbital STAR-37 motor burn, with a propellant mass $m^*_p = 700$ kg. Since it is assumed that the total number of particles is proportional to the propellant mass, Eq. (1) can be scaled for generic motor burn, with propellant mass of m_p as:

$$N_{>d} = N^*_{>d^*} \frac{m_p}{m^*_p} \left(\frac{d^*}{d} \right)^3 \tag{2}$$

The validity range of the above relation is from $d^* = 5$ mm to an upper limit of $d_{up} = 3$ cm. Since there is an observational evidence of slag particles smaller than 5 mm, an extension of Eq. (2) was devised. The extended distribution is given by:

$$N_{>d} = N^*_{>d^*} \frac{m_p}{m^*_p} \left(\frac{w + d^{*3}}{w + d^3} \right) \tag{3}$$

The smallest diameter for Eq. (3) is now $d_{lw\,x} = 10$ μm and, for continuity reasons, the upper limit is set equal to the lowest limit of Eq. (1), i.e., $d_{up\,x} = d^* = 5$ mm. Then w can be obtained from the relation:

$$w = \frac{\frac{N^*_{>d_{lw\,x}}}{N^*_{>d^*}} d^3_{lw\,x} - d^{*3}}{1 - \frac{N^*_{>d_{lw\,x}}}{N^*_{>d^*}}}$$

where the total number of objects is set to: $N^*_{>d_{lw\,x}} = 3 \cdot 10^7$ and the diameters are in mm.
The objects are supposed spherical. Within all the objects produced, 50 % are assumed to be Al_2O_3 slag, with an average density of 3.5 g/cm^3 and 50 % liner residuals, with an average density of 1.8 g/cm^3.
The particles are ejected with constant $\Delta V = 75$ m/s in a random direction opposite to the velocity vector, within a cone of 20° aperture (average nozzle half angle of kick stage motors) (Fig. 2).
The management of the slag releases is controlled inside the files *lclass_N.dat* (C.3). For each upper stage using solid propellant, the user has to set the value of the solid propellant mass, in kg. If the mass is set to 0, this means that the upper stage is not a solid propellant one (note that the solid rocked slag production can be altogether excluded by setting to 0 the propellant mass for each of the upper stages listed in the *lclass_N.dat* (C.3) file). Then it has to be specified if the motor is a perigee or an apogee kick motor, to calculate the location of the ejection.
Note that all the slag particles in the *historical population* (i.e. the slag already in orbit at the starting epoch of the simulation) are directly read from the initial population file (e.g. from the MASTER 2001 population).

2.3 The EVOLVE 4 breakup model

The new NASA's EVOLVE 4 breakup model was added to the SDM 3.0 program to simulate the generation of debris clouds produced by on-orbit explosions and collisions. The model has

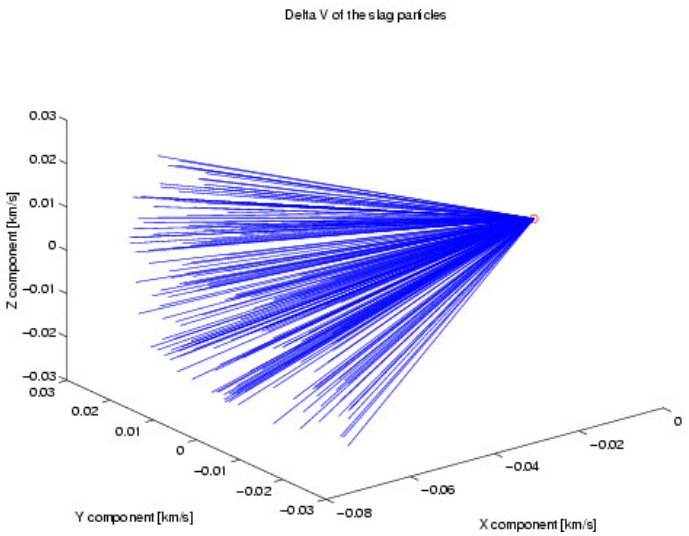

Figure 2: Ejection direction of the slag/liner particles for a sample solid rocket motor firing

been implemented using as reference the paper by Johnson *et al.* [2]. Further small changes with respect to this published model were made possible thanks to *personal communications* with P. Krisko, from NASA JSC.

The fundamental independent variable in EVOLVE 4.0 is the fragment's characteristic length (size) that is equivalent to the diameter of spherical objects. The size, which was derived from the radar cross section (RCS) of cataloged objects or obtained by direct measurements of small fragments in laboratory tests, was introduced in the new model through an empirical characteristic length distribution.

An area-to-mass (A/M) distribution was included as a function of the fragment's characteristic length, against the former model (EVOLVE 3.0) in which a single area-to-mass value corresponded to each given debris size. The combination of the fragment's area, obtained by applying the EVOLVE 4.0 empirical relationship for the size-to-area conversion, and the area-to-mass ratio yields the mass.

A new velocity distribution as a function of area-to-mass was included as well to compute the velocities imparted at breakup. The Δv distribution, like the area-to-mass distribution, was based on the analysis of the observed on-orbit and ground tests (in the first place the Satellite Orbital Debris Characterization Impact Test: SOCIT) breakup fragments.

The use of the EVOLVE breakup model is triggered by two flags in the input file *sdm.dat* (C.4). The flag $c6$ defines the mass distribution after a collision. If $c6 = q$ then the EVOLVE model is used. For the explosions, if the flag $ev\_exp = 1$ the EVOLVE model is used, otherwise the old SDM models are used (see the *User Manual* for details).

2.3.1 Size distribution

Since the size, or characteristic length L_c, is more directly related to both the on-orbit and laboratory test data, L_c was adopted as the basic independent variable in EVOLVE 4.0. Two distinct power law distributions for the number of fragments of a given size and larger were included in the new breakup model for explosions and collisions.

Explosions

The principal change in the new explosion model was that the distinction between the so-called low and high intensity explosions was abandoned. Only one size distribution was in fact included in EVOLVE 4.0 to compute the number of explosion fragments of size L_c or larger (in meters) using the following equation:

$$N(L_c) = S\,6\,L_c^{-1.6} \tag{4}$$

where S is the so-called scaling factor depending on the exploded object. S is a unit-less number which value is assigned on the basis of the average observed number of fragments produced by each type of explosion. Table 1 shows some scaling factors that are associated, in EVOLVE 4.0, to a set of historical breakups (P. Krisko, *personal communication*, 2003). In the SDM 3.0 the scaling factor S is specified by the user inside the file *expinp.dat* (C.1) for each class of explosion. For instance, if $S = 1$, 378 555 objects larger than 1 mm are created. Out of these, 9 515 are larger than 1 cm.

Exploded Object	Scaling Factor
Rocket bodies, excepting the specific case of the SOZ units	1.0
Ullage motors (SOZ) of the 4^{th} stage of the Russian Proton (SL-12) launch vehicle	0.1
Soviet ELINT Ocean Reconnaissance Satellites (EORSAT)	0.6
Soviet early warning satellites	0.1
Soviet battery-related events	0.5
Soviet Anti-Satellite (ASAT) events	0.3

Table 1: Scaling Factors for the EVOLVE 4.0 Explosion Events

Collisions

A collision between two space objects may be non-catastrophic, wherein a fragmentation of the smallest object along with a craterization of the largest object typically occurs, or catastrophic, if both objects are completely fragmented. A collision is catastrophic if $E_c/M_T \geq 40\,000$ J/kg, where E_c is the kinetic energy of the smallest object and M_T (in kg) is the mass of the largest object. The following power law distribution for the number of collision fragments of a given size and larger was implemented in the updated EVOLVE model:

$$N(L_c) = 0.1\,M^{0.75}\,L_c^{-1.71} \tag{5}$$

In Eq.5, L_c is in meters while the value of M depends on the type of collision. If a collision is catastrophic, then $M = M_t + m_p$, where m_p is the mass (in kg) of the smallest object (the projectile). On the contrary, for a non-catastrophic collision $M = m_p V_c$, where V_c is the collision velocity (in km/s). For instance, if $M_T = 1000$ kg, $m_p = 10$ kg and the impact velocity is 10 km/s, 2 416 835 objects larger than 1 mm are created. Out of these, 47 185 are larger than 1 cm.

2.3.2 Area to mass distributions

Two area-to-mass distribution functions have been developed and included in EVOLVE 4.0 for breakups: one for rocket bodies and another one for spacecraft. For rocket body (R/B) breakups, the rocket bodies distribution function is used for fragments larger than 11 cm, while the small fragment distribution function (from the SOCIT collisional tests) for fragments smaller than 1.7 cm. A similar distinction is made for spacecraft breakups, so that the spacecraft distribution function is used for fragments larger than 11 cm and the small fragment distribution function for fragments smaller than 8 cm. An additional function is used to bridge the gap in between 1.7 and 11 cm for rocket bodies and 8-11 cm for spacecraft (Tab.2). This function is defined in the following (P. Krisko, *personal communication*, 2003). For each characteristic length L_c inside the above gaps a new variable, named the bridging function (BF), is computed both for rocket bodies as:

$$BF = 10(log_{10} L_c + 1.76)$$

and for spacecraft as:

$$BF = 10(log_{10} L_c + 1.105) \ .$$

Afterwards, the bridging function is compared to a random number R in between 0 and 1. If $BF > R$ then the rocket bodies (or spacecraft) distribution is used for that L_c in the transition region. On the contrary, if $BF \leq R$, the area-to-mass distribution for small fragments (SOCIT) is applied.

	$L_c \leq 1.7$ cm	$1.7 < L_c < 11$ cm	$L_c \geq 11$
Rocket Bodies	Small fragments distribution	Bridging function	Rocket bodies distribution

	$L_c \leq 8$ cm	$8 < L_c < 11$ cm	$L_c \geq 11$
Spacecraft	Small fragments distribution	Bridging function	Spacecraft distribution

Table 2: Area-to-mass Distribution Functions for the EVOLVE 4.0 Breakup Model

In order to obtain a good fit to the data, the area-to-mass distributions for spacecraft and rocket bodies were modelled as the sum of two normal distributions in $log_{10}(A/M)$.

Rocket Bodies

The functional form for the rocket bodies fragments larger than 11 cm is expressed as:

$$D^{R/B}_{A/M}(\lambda_c,\chi) = \alpha^{R/B}(\lambda_c)N(\mu_1^{R/B}(\lambda_c),\sigma_1^{R/B}(\lambda_c),\chi) + (1-\alpha^{R/B}(\lambda_c))N(\mu_2^{R/B}(\lambda_c),\sigma_2^{R/B}(\lambda_c),\chi)\ , \tag{6}$$

where:

$$\begin{aligned} \lambda_c &= log_{10}(L_c) \\ \chi &= log_{10}(A/M) \quad \text{is the variable in the distribution} \end{aligned} \tag{7}$$

and N is a normal distribution function with mean μ and standard deviation σ:

$$N(\mu,\sigma,\chi) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{8}$$

The amplitude, $\alpha^{R/B}(\lambda_c)$, is in the range of 0 and 1, so that $D^{A/M}_{R/B}$ is itself normalized. The functional forms of the five parameters $\alpha^{R/B}$, $\mu_1^{R/B}$, $\sigma_1^{R/B}$, $\mu_2^{R/B}$,$\sigma_2^{R/B}$, are:

$$\alpha^{R/B} = \begin{cases} 1 & \text{if } \lambda_c \le -1.4 \\ 1 - 0.3571(\lambda_c + 1.4) & \text{if } -1.4 < \lambda_c < 0 \\ 0.5 & \text{if } \lambda_c \ge 0 \end{cases}$$

$$\mu_1^{R/B} = \begin{cases} -0.45 & \text{if } \lambda_c \le -0.5 \\ -0.45 - 0.9(\lambda_c + 0.5) & \text{if } -0.5 < \lambda_c < 0 \\ -0.9 & \text{if } \lambda_c \ge 0 \end{cases}$$

$$\sigma_1^{R/B} = 0.55$$

$$\mu_2^{R/B} = -0.9$$

$$\sigma_2^{R/B} = \begin{cases} 0.28 & \text{if } \lambda_c \le -1.0 \\ 0.28 - 0.16(\lambda_c + 1) & \text{if } -1 < \lambda_c < 0.1 \\ 0.1 & \text{if } \lambda_c \ge 0.1 \end{cases}$$

Spacecraft

The functional form for the spacecraft breakup fragments is expressed as:

$$D^{S/C}_{A/M}(\lambda_c,\chi) = \alpha^{S/C}(\lambda_c)N(\mu_1^{S/C}(\lambda_c),\sigma_1^{S/C}(\lambda_c),\chi) + (1-\alpha^{S/C}(\lambda_c))N(\mu_2^{S/C}(\lambda_c),\sigma_2^{S/C}(\lambda_c),\chi)\ ,$$

where λ_c , c and N are defined as in Eqs. 7 and 8, while the distribution parameters are computed as follows:

$$\alpha^{S/C} = \begin{cases} 0 & \text{if } \lambda_c \le -1.95 \\ 0.3 + 0.4(\lambda_c + 1.2) & \text{if } -1.95 < \lambda_c < 0.55 \\ 1 & \text{if } \lambda_c \ge 055 \end{cases}$$

$$\mu_1^{S/C} = \begin{cases} -0.6 & \text{if } \lambda_c \leq -1.1 \\ -0.6 - 0.318(\lambda_c + 1.1) & \text{if } -1.1 < \lambda_c < 0 \\ -0.95 & \text{if } \lambda_c \geq 0 \end{cases}$$

$$\sigma_1^{S/C} = \begin{cases} 0.1 & \text{if } \lambda_c \leq -1.3 \\ 0.1 + 0.2(\lambda_c + 1.3) & \text{if } -1.3 < \lambda_c < -0.3 \\ 0.3 & \text{if } \lambda_c \geq -0.3 \end{cases}$$

$$\mu_2^{S/C} = \begin{cases} -1.2 & \text{if } \lambda_c \leq -0.7 \\ -1.2 - 1.333(\lambda_c + 0.7) & \text{if } -0.7 < \lambda_c < -0.1 \\ -2.0 & \text{if } \lambda_c \geq -0.1 \end{cases}$$

$$\sigma_2^{S/C} = \begin{cases} 0.5 & \text{if } \lambda_c \leq -0.5 \\ 0.5 - (\lambda_c + 0.5) & \text{if } -0.5 < \lambda_c < -0.3 \\ 0.3 & \text{if } \lambda_c \geq -0.3 \end{cases}$$

Small fragments

For small fragments generated in laboratory experiments, it was found that only one normal distribution is required to provide a fit to the data. As a result, a single area-to-mass distribution function was derived from the SOCIT test fragments for both rocket bodies and spacecraft as follows:

$$D_{A/M}^{SOC}(\lambda_c, \chi) = N(\mu^{SOC}(\lambda_c), \sigma^{SOC}(\lambda_c), \chi) \ ,$$

with λ_c , c and N defined as in Eqs. 7 and 8, and the distribution parameters given by:

$$\mu^{SOC} = \begin{cases} -0.3 & \text{if } \lambda_c \leq -1.75 \\ -0.3 - 1.4(\lambda_c + 1.75) & \text{if } -1.75 < \lambda_c < -1.25 \\ -1.0 & \text{if } \lambda_c \geq -1.25 \end{cases}$$

$$\sigma^{SOC} = \begin{cases} 0.2 & \text{if } \lambda_c \leq -3.5 \\ 0.2 + 0.1333(\lambda_c + 3.5) & \text{if } \lambda_c > -3.5 \end{cases}$$

2.3.3 Average cross sectional area

The average cross-sectional area A_x was modelled as having a one-to-one correspondence with the characteristic length L_c. The following relationship for the size to area conversion was derived:

$$\begin{aligned} A_x &= 0.540424\ L_c^2 && \text{if } \mathrm{L_c} < 0.00167 \text{ m} \\ A_x &= 0.556945\ L_c^{2.0047077} && \text{if } \mathrm{L_c} \geq 0.00167 \text{ m} \end{aligned}$$

2.3.4 Mass

For a given fragment, the mass M is simply determined from A/M for that fragment and the fragment's average cross sectional area A_x via the relationship:

$$M = \frac{A_x}{A/M}$$

2.3.5 Delta velocity distributions

A new ΔV distribution function as a function of area-to-mass was derived to fit the observed on-orbit breakup data and the SOCIT tests data. The ΔV values were modelled as a normal distribution in $log_{10}(\Delta V)$ about the mean value μ with a standard deviation of σ.
The ΔV distribution function of fragments from explosions is:

$$D_{\Delta V}^{EXP}(\chi, \nu) = N(\mu^{EXP}(\chi), \sigma^{EXP}(\chi), \nu) \ ,$$

where:

$$\begin{aligned} \nu &= log_{10}(\Delta V) \\ \chi &= log_{10}(A/M) \\ \mu^{EXP} &= 0.2\chi + 1.85 \\ \sigma^{EXP} &= 0.4 \ . \end{aligned}$$

A similar ΔV functional form is used for collisions:

$$D_{\Delta V}^{COL}(\chi, \nu) = N(\mu^{COL}(\chi), \sigma^{COL}(\chi), \nu) \ ,$$

where:

$$\begin{aligned} \nu &= log_{10}(\Delta V) \\ \chi &= log_{10}(A/M) \\ \mu^{COL} &= 0.9\chi + 2.9 \\ \sigma^{COL} &= 0.4 \, . \end{aligned}$$

2.3.6 Implementation details of the EVOLVE breakup model inside SDM 3.0

In this section some details about the actual implementation of the EVOLVE model inside SDM 3.0 is given. In particular the points that are not clarified or different from what reported in [2] are described.
The first important point to stress is that the EVOLVE model has been developed, and applied so far, for particles larger than 1 cm. Since the range of applicability of SDM 3.0 starts at 1 mg, we implemented an extension of the EVOLVE model down to the limiting size $L_c > 1$ mm. This was deemed acceptable since from P. Krisko, (*personal communication*, 2003) we have been made aware of the fact that NASA is starting to use and validate the model down to 1 mm.
Also the extraction of the objects from the size distribution are slightly different from what reported in [2].
In the case of explosions the procedure is the following (P. Krisko, *personal communication*, 2003):

1. From Eq. 4 the number of objects larger than 1 mm and smaller that 1 m are calculated and extracted from the proper distribution of size;

2. From the size and the area-to-mass relation the mass of each fragment is calculated;

3. If the sum of all the masses, M_{sum} does not add up to the mass of the exploding object M_{exp} a number $0 < N_{big} \leq 8$ is randomly extracted. The N_{big} fragments are produced with the following criteria:

   - if $N_{big} = 1$ then all the mass $M_{exp} - M_{sum}$ goes into one fragment;
   - if $N_{big} = 2$ then one fragment with mass $0.55\,(M_{exp} - M_{sum}) \leq M_1 < 0.65\,(M_{exp} - M_{sum})$ and one fragment with mass $M_2 = M_{exp} - M_{sum} - M_1$ are generated;
   - if $N_{big} \geq 3$ then one fragment with mass $0.55\,(M_{exp} - M_{sum}) \leq M_1 < 0.65\,(M_{exp} - M_{sum})$, one fragment with mass $0.2\,(M_{exp} - M_{sum} - M_1) \leq M_2 < 0.3\,(M_{exp} - M_{sum} - M_1)$ are produced. Then the rest of the mass is equally divided in the remaining $N_{big} - 2$ fragments.

4. Since these large fragments are generated using the mass instead of the length L_c, the size L_C and the area A of a fragment with mass M is obtained using the following relations:

$$A/M = 0.3\,M^{-0.43}$$

$$A = \frac{A/M}{M}$$

$$L_c = \sqrt{\frac{4A}{\pi}}$$

According to NASA, the above method is based on observations of large pieces of on-orbit breakups.
For catastrophic collisions, the fragments, even larger than 1 m, are extracted from the size distribution (Eq. 5) until the total mass is reached. If the last mass pulled goes over the mass limit, the last mass is reduced to match the total mass and the above method is used to estimate A/M and L_c.
For non-catastrophic collisions, the ejecta mass is extracted from the size distribution (Eq. 5) until the total mass is reached. If the last mass pulled goes over the mass limit, the last mass is reduced to match the total mass and the above method is used to estimate A/M and L_c.
In Figures 3 and 4 a sample of the distribution of particles produced by different fragmentation events is shown. In Fig. 3 the cumulative distribution of L_c is shown for the explosion of a 1000 kg spacecraft. The scaling factor S of Eq. 1 is set to 0.1. In Fig. 3 (right panel) the cumulative distribution of L_c is shown for the collision between a 1000 kg spacecraft, and a 10 kg projectile at the impact speed of 10 km/s. Finally, in Fig. 4 the cumulative mass distribution of fragments (with mass larger than 1 g) is shown, for the same collision as above, generated with the EVOLVE 4 breakup model and the CNUCE collision breakup model [4] [5].

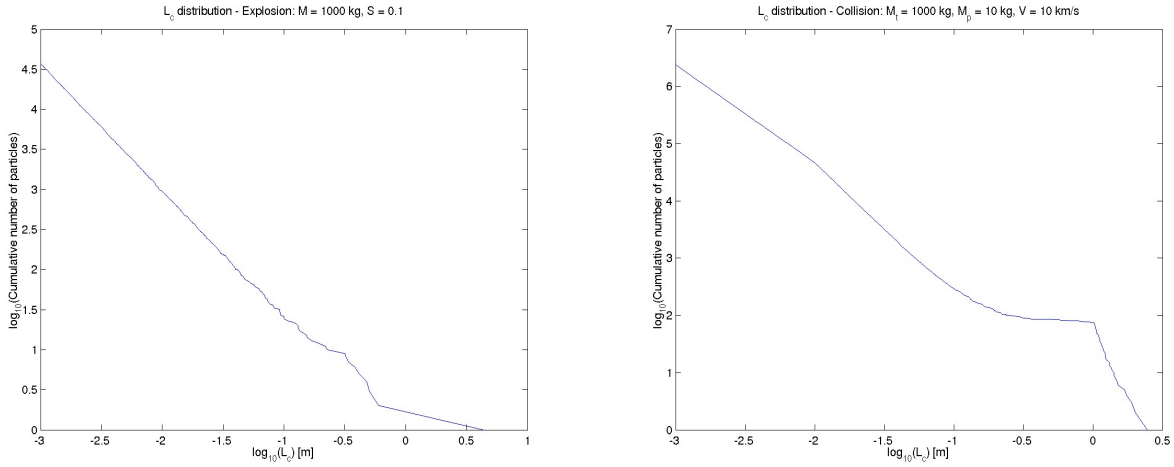

Figure 3: Cumulative distribution of characteristic length for the explosion of a 1000 kg spacecraft ($S = 0.1$) (left panel) and for a collision between a 1000 kg spacecraft, and a 10 kg projectile at 10 km/s (right panel), with the EVOLVE 4 breakup model.

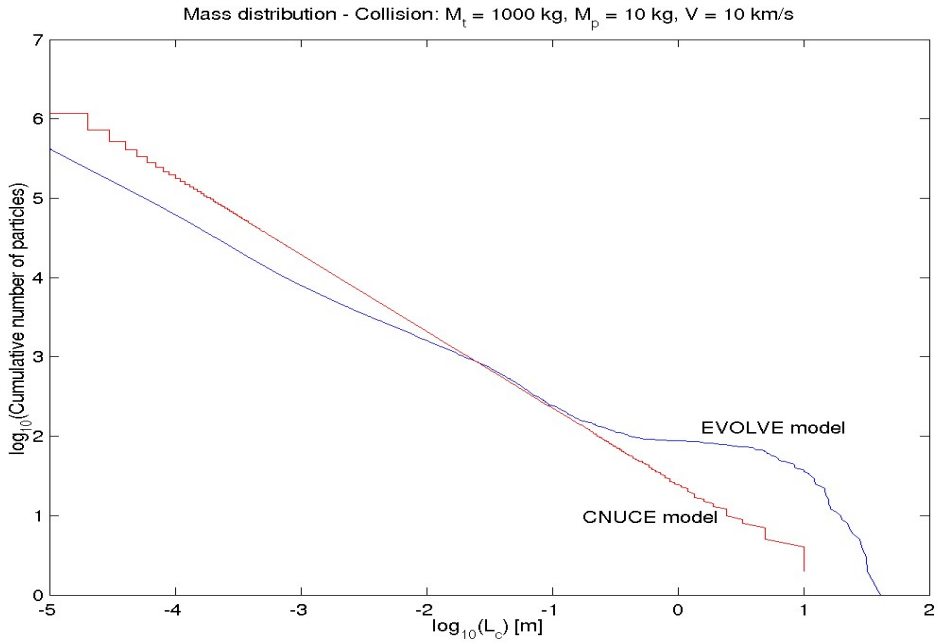

Figure 4: Cumulative mass distribution for a collision between a 1000 kg spacecraft, and a 10 kg projectile at 10 km/s with the EVOLVE 4 breakup model (blue line) and with the CNUCE breakup model (red line).

2.4 Low velocity collision model

The low velocity collision model developed by T. Hanada has been implemented in SDM 3.0, following the specifications given in [3]. The model should account for collisions in the GEO region where the impact velocity are considerably lower than the typical velocities involved in LEO collisions.
The cumulative number of fragments of mass larger than m is given by:

$$y = 0.782(m/m_e)^{-0.679} \tag{9}$$

where m_e is the ejecta mass given by:

$$m_e = m_p V_i^2$$

where m_p is the projectile mass in kg, and V_i is the impact velocity in km/s.
If the NASA breakup model of EVOLVE 4 (see Sec. 2.3) is selected by the user (by setting the flag *c6* = *q* in the file *sdm.dat*), then a slightly different distribution is adopted, in terms of the characteristic length L_c, measured in meters:

$$N(L_c) = 0.39(m_e)^{-0.62} L_c^{1.62}\,. \tag{10}$$

This model is used only when the impact velocity is below a limit set in the header file *hanada.h* (A.6). Presently a default of 5 km/s is set (see [3]).
Anyway, disregarding the impact velocity, the user can decide to use or not the model by setting the flag *hanada* in the input file *sdm.dat* (C.4). If this flag is set to 0 the model is never used, independently from the impact velocity.

2.5 Mitigation measures

Several mitigation options can be simulated with SDM. A brief summary of some of the mitigation options, with particular emphasis to the new features of SDM 3.0, is given below.
The mitigation options are controlled by the user mainly within the input file *sdm.dat* (C.4).
For satellites in the geostationary region the mitigation option is implemented in the following way. Each new GEO satellite injected in orbit is labeled (in the input file *lclass_N.dat* (C.3)) with its own operational lifetime. At the end of the lifetime the spacecraft can be re-orbited in a super-GEO disposal zone, approximately 300 km above the ring, with the exact altitude depending from the actual cross-sectional area of the spacecraft. As a default, the IADC formula for the altitude h (in km) of the storage orbit is implemented:

$$h = 235 + (1000 \cdot C_r \cdot A/M) \tag{11}$$

where C_r is the solar radiation pressure coefficient and A/M is the area-to-mass ratio of the spacecraft. The ΔV required by the re-orbiting manoeuvre (in km/s) is recorded in the output file *deltav.dat* (see [7]). The actual re-orbiting altitude is extracted from a triangular distribution, centered on h, whose upper and lower limits are set by the user in the file *sdm.dat* (C.4).

The user can choose not to re-orbit a given class of GEO satellites and this can be done by assigning it a lifetime longer than the total simulation time span in the file *lclass_N.dat* (C.3). For LEO satellites there are a few options. The spacecraft can be de-orbited to an orbit with a given residual lifetime, under the effect of the air drag. The residual lifetime for all the satellites, for all the simulation time span is given by the user in the input file *sdm.dat* (C.4). A residual lifetime of 0 years means immediate de-orbiting at end-of-life. The de-orbiting can be to a circular disposal orbit (Fig. 5, left panel). An impulsive manoeuvre is simulated, with a minimum energy transfer between coplanar orbits (i.e. Hohmann like transfers and no changes of inclination). The de-orbiting to a circular orbit might have some advantages from the point of view of collision avoidance with spacecraft orbiting in LEO. On the other hand it is very expensive in terms of ΔV, if performed with impulsive manoeuvres [7]. Therefore the user can decide to de-orbit to an elliptical orbit (Fig. 5, right panel). This requires a single manoeuvre to lower the perigee, thus saving propellant. In this latter case the perigee altitude of the disposal orbit is found by a fast numerical (bisection) method which does not increase significantly (i.e. less than 1.5 %) the CPU time over a typical single run of SDM. In both the cases (circular and elliptic), the ΔV required by the de-orbiting manoeuvre (in km/s) is recorded in the file *deltav.dat*.
In SDM 3.0 the possibility to re-orbit the satellites at end-of-life in a super-LEO storage orbit (with a given altitude and width) is foreseen. The software can operate in a fixed or in a ΔV optimizing way. The user can decide that all the satellites with perigee above a given altitude (given by the flag *perdeo* in the input file *sdm.dat* (C.4)) have to be re-orbited into the storage orbit. Otherwise the software automatically performs the most efficient manoeuvre, in terms of ΔV, between de-orbiting and re-orbiting. This choice is controlled by the flag *au_fi* in the input file *sdm.dat* (C.4). The altitude of the storage orbit is given by the parameter *deoalt* in the input file *sdm.dat* (C.4). Like in the GEO case, the actual re-orbiting altitude is extracted from a triangular distribution, centered on *deoalt*, whose upper and lower limits are set by the user in the file *sdm.dat* (C.4) (the same values are used both for the LEO and GEO disposal zones). Note that these limits also set the width of the disposal zone, i.e., the width of the zone is equal to the sum of the lower and upper limit values. Again, the ΔV required by the re-orbiting manoeuvre (in km/s) is recorded in the file *deltav.dat* (see Fig. 6).

2.5.1 Deorbiting of the constellation satellites

A dedicated mitigation strategy was implemented to handle the constellations of satellites. The deorbiting/reorbiting of old constellation satellites are handled in two different ways. The disposal policy of each constellation is defined by the flag *deo_pol* in the file *launch.in* (C.2). If $deo\_pol = 0$ the satellites are disposed in the same way as the normal satellites. I.e., they follow the general rule introduced in the file *sdm.dat* (C.4) and described in Chap. 2.5. Otherwise if $deo\_pol = 1$ the satellites are moved to the disposal orbit specified by the orbital elements given in the vector *deo_ele* defined in the input file *launch.in* (C.2). The deorbiting actions are coded in the subroutine *conste* (A.5) (for what concerns the launches and replacement of

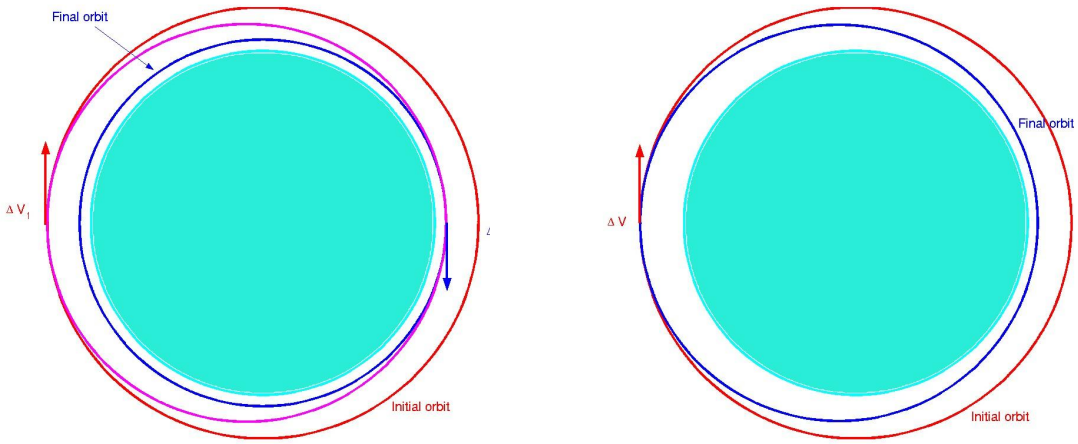

Figure 5: Scheme of the de-orbiting into a circular (left) and into an elliptic (right) orbit with a given residual lifetime.

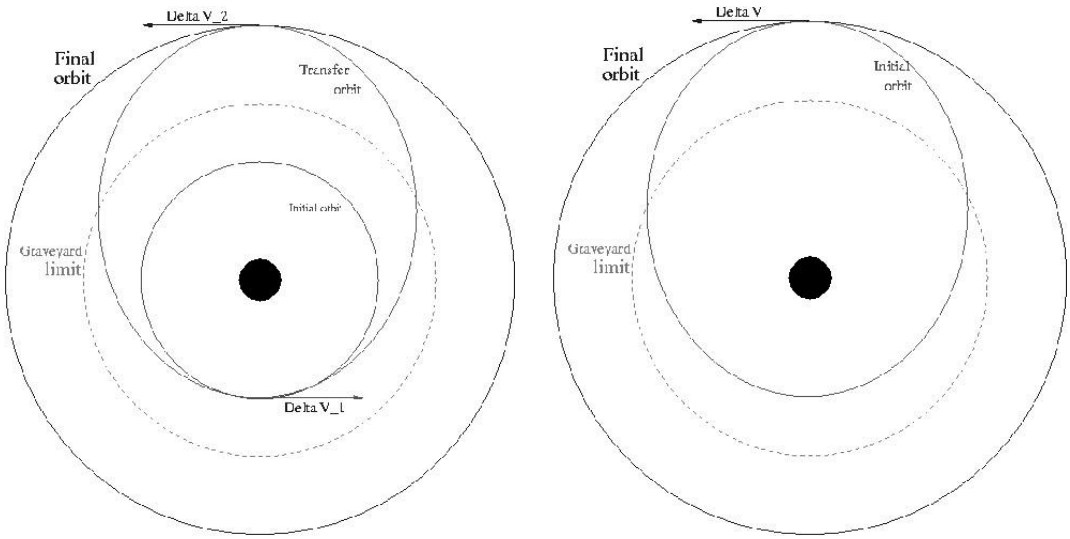

Figure 6: Scheme of the re-orbiting into a storage orbit with a two-burn (left) and with a single-burn (right) manoeuvre.

satellites during the constellation life) and in *chlife* (A.1) for the elimination of the constellation at end-of-life.

2.6 Miscellaneous changes

In this section some other changes introduced in SDM 3.0 are described.

2.6.1 Time dependent launch traffic

In order to simulate a time varying launch traffic a new feature was added to SDM 3.0. In the old versions of SDM a single file *lclass.dat* contained the details of the launch traffic in terms of classes of objects launched. In SDM 3.0 the time varying launch traffic is controlled by the flag *fl_lcla* inside the file *launch.in* (C.2). If *fl_lcla* = 0 the code looks for a single file with the launch traffic details, named *lclass_1.dat* (C.3). Otherwise, if *fl_lcla* = 1, the code looks for a launch traffic file for every decade of the simulation interval. E.g., if the time span of the run is 50 years the user has to prepare five *lclass_N.dat* (C.3) with N=1,2,3,4,5. The files *lclass_N.dat* (C.3) shall be placed in the SDM input directory and then the macro *sdm* (B.2) copies them into the execution directory.

2.6.2 Representative mass of the colliding objects

In SDM 3.0 the representative object for each mass bin, to be used in the collision rate calculation (in the routine *colgen.f* (A.2)), has the average mass of all the objects with mass falling inside the mass bin limits, included in MASTER 2001. Along with the mean value also the standard deviation is recorded, but presently it is not used because the distribution of the masses inside the bin limits is not at all flat (or Gaussian) and ad hoc distributions should be studied for the different bins. This work is at present deemed unnecessary.
The mean values are stored in the header file *colobj.h* (A.7). The mean values have been calculated, for the three largest mass bins, for LEO ($0 < a < 10\,000$ km), MEO ($10\,000 < a < 40\,000$ km) and GEO ($a > 40\,000$ km) spacecraft. The values presently used, for the 10 mass bins, are listed in Tab. 3.
Note that only for the three largest bins a different value has been chosen for the various orbital regimes. The different values are used according to the altitude shell in which the collision is happening.

2.6.3 Objects sampling

As described in Chap. 1, to keep the number of objects within a manageable value throughout a run of SDM, the objects below a given mass threshold are sampled. In SDM the objects are divided in *nbinm* mass bins (*nbinm* = 10 as a default) as specified in *parbin.h* (A.14). Within the code, another parameter, called *nsmall*, indicates the number of bins with small objects, i.e., objects that are sampled in the various source mechanisms. Differently from SDM 2.0, the value of *nsmall* is now equal to the number of the bin for which the sampling factor is larger than 1. This means that the user has not anymore to set this value which is handled automatically by the code. Moreover, the variable *ammin* (set in the header file *datexp.h* (A.9)), which gives the limit mass below which the objects are treated as distributions in the explosion

Mass bin	LEO	MEO	GEO
1	2.07×10^{-6}	2.07×10^{-6}	2.07×10^{-6}
2	2.46×10^{-5}	2.46×10^{-5}	2.46×10^{-5}
3	2.68×10^{-4}	2.68×10^{-4}	2.68×10^{-4}
4	0.0028	0.0028	0.0028
5	0.026	0.026	0.026
6	0.299	0.299	0.299
7	3.482	3.482	3.482
8	39	55	37
9	479	607	484
10	2148	1741	2076

Table 3: Mass, in kg, of the representative object to be used in the collision rate calculations, for each of the 10 mass bins, in the different orbital regimes.

and collision process, is checked by the code and set equal to the upper limit of the last sampled bin.

2.6.4 Running SDM from a CD ROM

To avoid storing the *historical population* density files in the hard disk, SDM 3.0 can be run also reading the density files from a CD ROM. This is done by using the macro *cd_sdm* (B.3) instead of the *sdm* macro. The macro *cd_sdm* mounts the CD ROM drive and sets the flag $cd\_hd = 1$. Therefore the code looks for the path of the density files in the header files *dirdeh_cd.h* (A.11) and *dirder_cd.h* (A.13) (instead of looking for the similar files *dirdeh.h* (A.10) and *dirder.h* (A.12) with the hard disk path).

2.6.5 Printout of the running population

The possibility to dump the whole running population on a file at epochs specified by the user was added. The years in which the running population shall be printed are specified in input in the file *snapsyears.dat* (C.5). In the file the user has to input either all the epochs for which the population is desired (one year per line) or a single line with the string "55555" which means that the running population has to be dumped each year. If no line is present in the file, then the running population is dumped only at the final epoch (default setting). The output files with the running population are named *running.yyyy* where *yyyy* is the epoch of the file.

3 The Post-Processing

The post-processing of SDM 3.0 was completely revised.
The main UNIX shell script to control all the post-processing is called *sdm_pos*. It allows the post-processing of the files contained in an output directory, selected by the user. The shell script works both on the averaged directory (created by *sdm_average*) or on a single output directory (created by *sdm_mult*). It allows the following actions:

1) VIEW PLOTS OF EVENTS (launches, explosions, collisions)
2) VIEW PLOTS OF NUMBER OF OBJECTS GLOBAL
3) VIEW PLOTS OF NUMBER OF OBJECTS DIVIDED IN CLASSES
4) VIEW PLOT OF (COLLISIONAL FRAGMENTS)/(TOTAL NUMBER)
5) VIEW PLOTS OF DENSITIES
6) VIEW PLOTS OF FLUXES
7) VIEW PLOTS OF COLLISION PROBABILITIES
8) VIEW PLOTS OF CRITICAL DENSITY
9) VIEW PLOTS OF RUNNING POPULATION IN THE ORBITAL ELEMENT SPACE
10) VIEW PLOTS OF RUNNING POPULATION IN THE "Imp. Vel vs. a" PLANE
11) VIEW PLOTS OF RUNNING POPULATION IN a - cos i PLANE
12) VIEW HISTOGRAMS OF THE DISTRIBUTION OF THE COLLISIONS
13) CHANGE WORKING DIRECTORY

By choosing a number from 1 to 12 the user is guided by a series of questions to the desired final plot. The option number 13 just changes the directory whose output is being analyzed.
All the plots shown on the screen are stored as post-script files in the selected output directory.
The Options 1 – 8 were already present in SDM 2.0 and will not be described in this report. They will be anyway resumed in the new User Manual. Here the Options 9 – 12 will be briefly described.

3.1 View plots of the running population in the orbital element space

Option 9 displays the plots of the *running population* in the orbital element space. Since the running population is printed only on selected years (see Sec. 2.6.5), first the code will display which running population files are available. The user has to select one of the available years. Then the choice is between three different representations of the *running population*:

1) semi-major axis vs eccentricity

2) semi-major axis vs inclination

3) eccentricity vs inclination

Then the user has to select a lower threshold, either in diameter (in meters) or in mass (in kg), for the objects to be displayed.
Once the choices are entered, the selected plot is shown on the screen and the corresponding postscript file is generated.

3.2 View plots of the running population in the "Imp. Vel vs. a" plane

In [8] a new representation of the space debris population was proposed. It allows a practical visualization of the debris population in terms of its impact velocity with respect to a given target, assumed in a circular orbit. Option 10, implements this method and allows the visualization of the *running population* in terms of its impact velocity (in km/s) with respect to a target on a given circular orbit.
Since the running population is printed only on selected years (see Sec. 2.6.5), first the code will display which running population files are available. The user has to select one of the available years. Then the orbit of the target has to be specified by its semimajor axis (in km), inclination (in degrees) and argument of the node (in degrees) (the eccentricity has to be zero). Then the user has to select a lower threshold, either in diameter (in meters) or in mass (in kg), for the objects to be displayed.
Once the choices have been done, the selected plot is shown on the screen and the corresponding postscript file is generated. The meaning of the lines shown in the plots are explained in details in [8] but, due to the novelty of the representation, it seems appropriate to recall it here, with the help of Fig. 7. The target object is located at the point with coordinated (0,-1) in the plot. The curved lines departing from this point (blue dashed lines) represent the tangency conditions for $I = 0°$. Remember that here I indicates the relative inclination between the target and the projectile orbital planes, and not simply the equatorial inclination of the projectile orbit. On the right low corner of the plot, the dashed blue line represent the tangency conditions for $I = 180°$. Along these curves, the orbit of the projectile is tangent at perigee (for $\mathrm{a_{target}/a} > -1$) or at apogee (for $\mathrm{a_{target}/a} < -1$) to that of the target. All the orbits between the two tangency conditions cross that of the target, i.e. the objects laying in this region are potential impactors. The straight line from $(9, -1.5)$ to about $(12, 0)$ (solid red line) is the locus of orbits with $\cos I = 0$. This means, e.g., that, if the target is in equatorial orbit this line is surrounded by the objects in polar orbit. The straight line from $(0, -1)$ to $(7, 0)$ separates projectiles whose velocity relative to the target has a positive component along the target's velocity vector from those having a negative one. In practice, projectiles with orbits falling on this line have an equal probability of hitting the target on the front or on the rear side; above this line impacts from the back predominate (the more so as the tangency condition is approached), and below it the opposite holds. Finally, the curve on the extreme right (magenta line) represents the condition $e = 0$ (*circular limit*) for $I = 180°$, limiting a forbidden region (corresponding to $e^2 < 0$), at its right. Note that only the positions of the two straight lines do not depend on the projectile's inclination (the first in a trivial way, as it is the locus of all orbits with $I = 90°$). All the other conditions are inclination dependent.

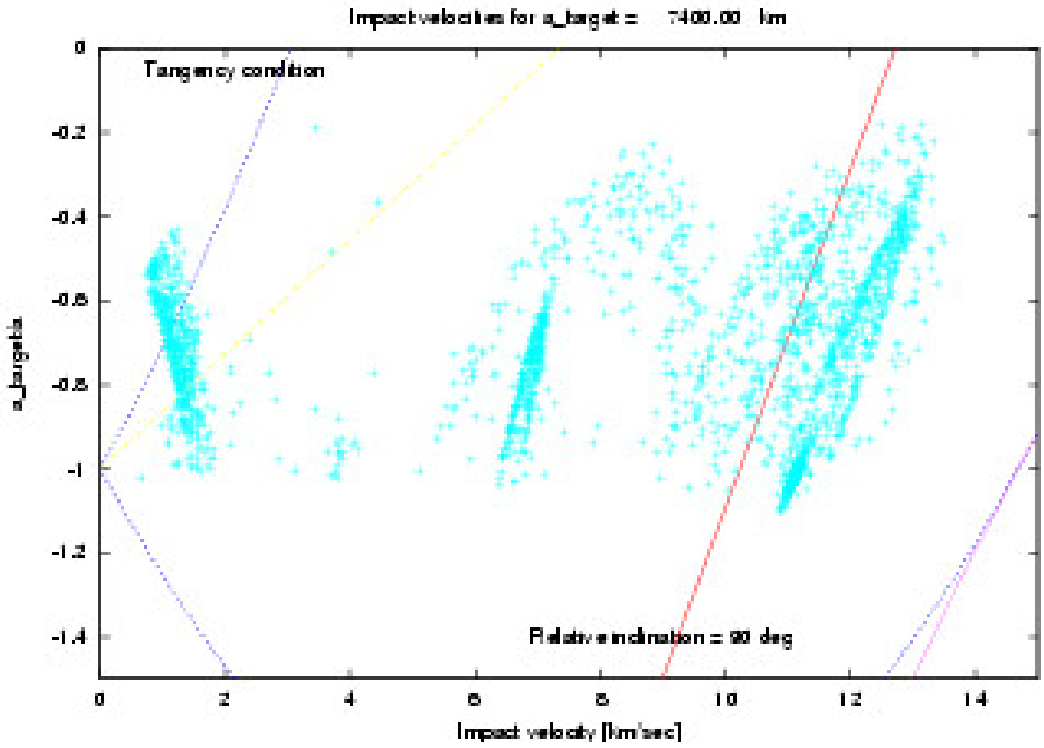

Figure 7: Representation of a population of debris in the "Impact velocity vs. semimajor axis" plane. See text for details.

3.3 View plots of the running population in the "a vs. cos I" plane

The method described in [8] allows also a practical visualization of the *intrinsic collision probability*, P, of a given debris population with respect to a target in a selected circular orbit. P is a quantity depending only on the two sets of orbital elements of the projectile and of the target. P may be interpreted as the collision rate between two bodies for which $(r + R) = 1\ m$. Of course, $P = 0$ if the two orbits cannot intersect each other for geometrical reasons; this occurs when the aphelion distance of the inner orbit is smaller than the perihelion distance of the outer one. Option 11, implements the method of [8] and allows the visualization of the *running population* in the $\cos I$ vs. a_{target}/a space. In this representation it is possible to visualize the value of intrinsic collision probability for each projectile in the population.

Since the running population is printed only on selected years (see Sec. 2.6.5), first the code will display which running population files are available. The user has to select one of the available years. Then the orbit of the target has to be specified by its semimajor axis (in km), inclination (in degrees) and argument of the node (in degrees) (the eccentricity has to be zero). Then the user has to select a lower threshold, either in diameter (in meters) or in mass (in kg), for the objects to be displayed.

Once the choices have been done, the selected plot is shown on the screen and the corresponding

postscript file is generated.

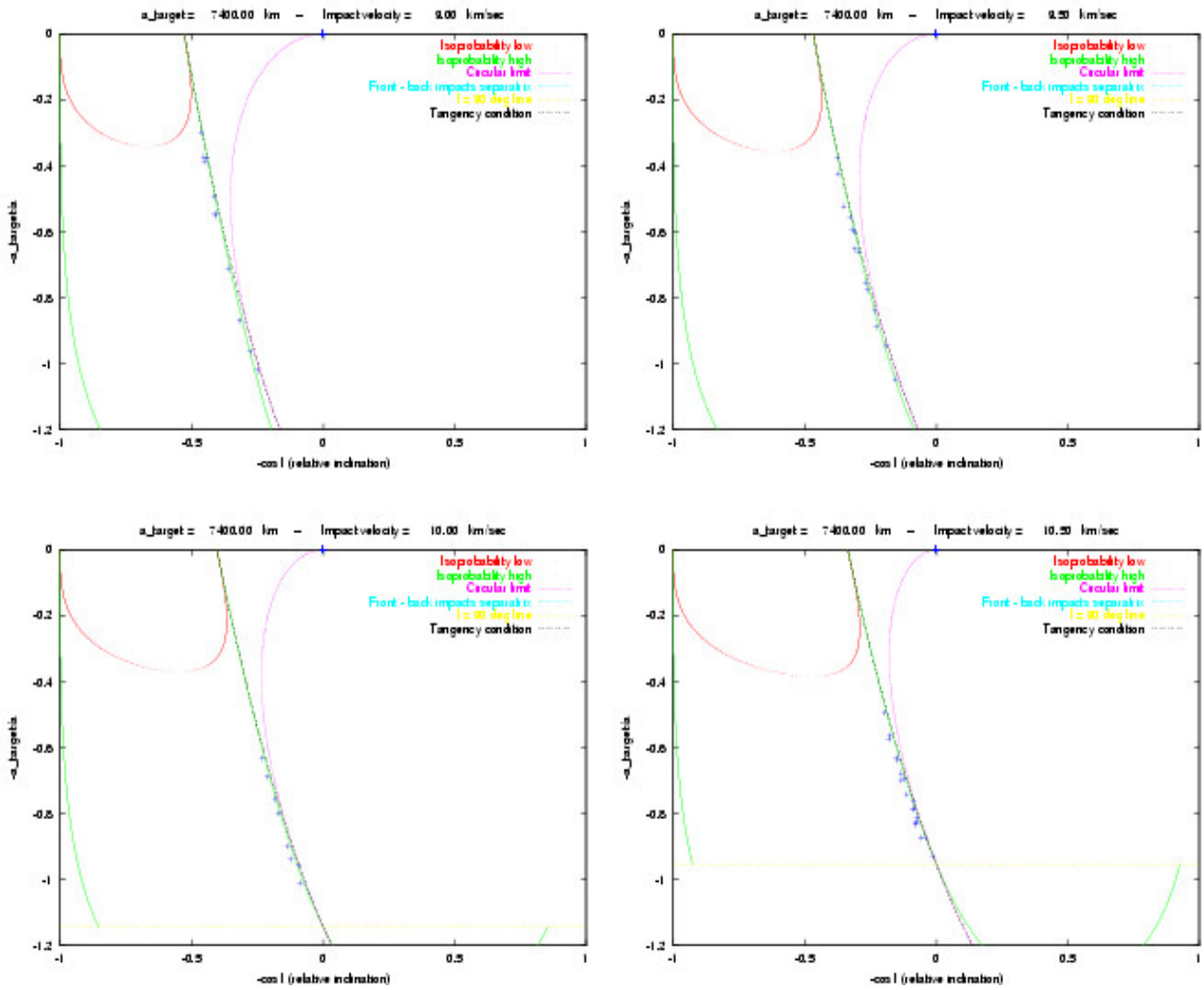

Figure 8: Representation of a population of debris in the "cos I vs. semimajor axis" plane. See text for details.

The meaning of the lines shown in the plots are explained in details in [8] but, due to the novelty of the representation, it seems appropriate to recall it here, with the help of Fig. 7 and Figs. 8–9. Each plot in Figs. 8 and 9 represent a "slice" along the third dimension of Fig. 7, i.e., a slice perpendicular to the plane of Fig. 7, taken at a given value of the impact velocity. Therefore, the user is asked to choose the position of the initial slice along the impact velocity axis, in km/sec. Then eight slices are shown at increasing values of the impact velocity, spaced from the initial one by an amount (in km/sec) given by the user.
Analyzing the geometry of the first panel in Fig. 8, we note first the magenta curve starting from (0,0). It represents the circular limit ($e = 0$) for zero inclination of the projectile ($I = 0°$);

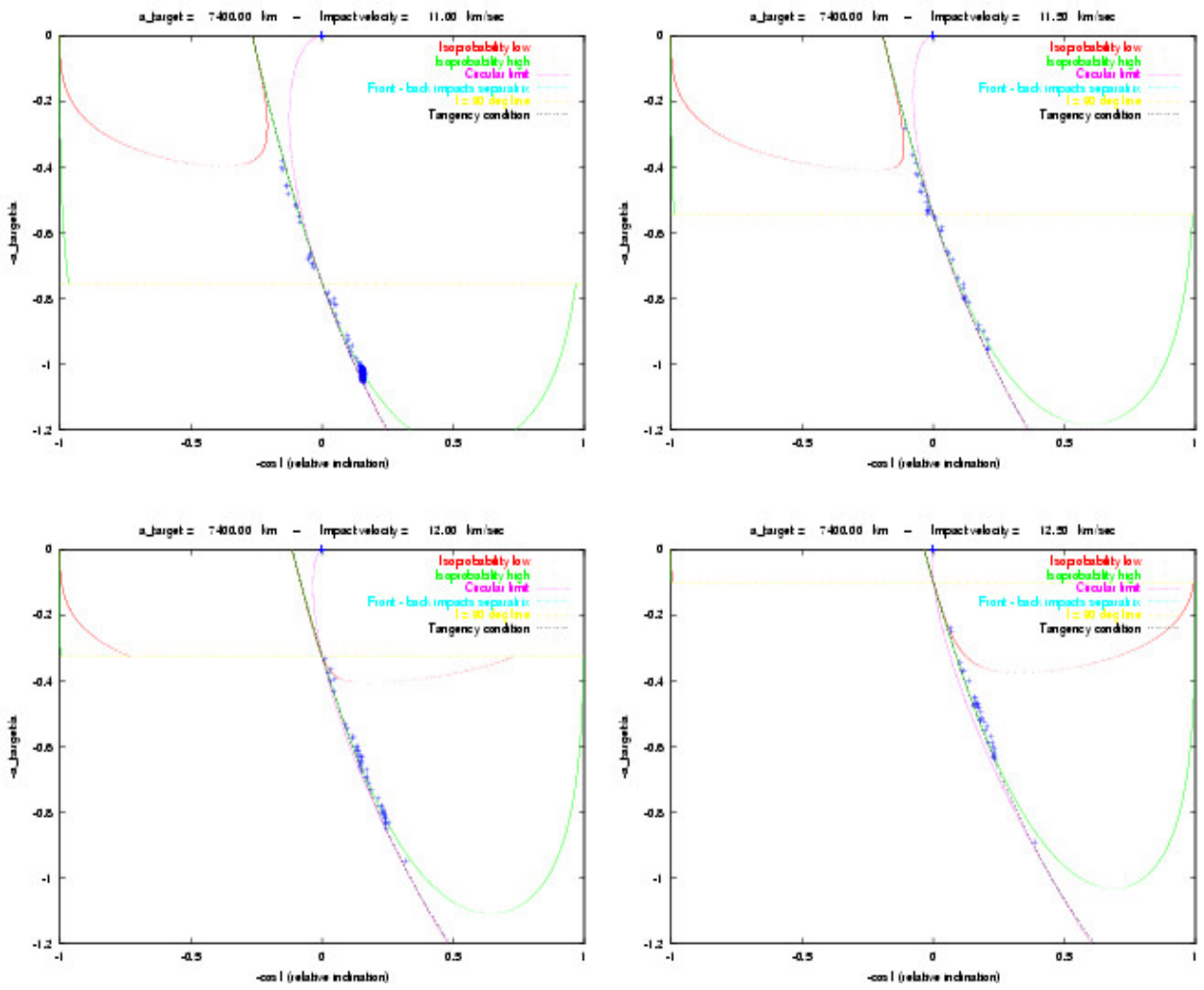

Figure 9: Representation of a population of debris in the "cos I vs. semimajor axis" plane. See text for details.

the region at its right is a forbidden one, as there projectile orbits would have $e^2 < 0$. The black dash-dotted line, always tangent to the the circular limit at $\mathrm{a_{target}/a} = -1$ is the tangency condition, already plotted in Fig. 7. Then two curves, green and red are plotted. These are curves of constant intrinsic probability of collision P. To draw these two curves, the user is asked the desired values of the probabilities (in $\mathrm{m^2\ yr^{-1}}$), a lower and a higher value, displayed by the red and the green curve, respectively. E.g., in Figs. 8 and 9, a lower value of $P_1 = 10^{-11}$ and an upper value of $P_2 = 10^{-10}$ are shown. Excluding the first two panels of Fig. 8, a horizontal yellow line is also present; this is the condition $\sqrt{a/a_0(1-e^2)}\cos I = 0$, that is fulfilled either for $I = 90°$ or for $e = 1$ (rectilinear orbits).
In the first panel of Fig. 8 the forbidden region ($e^2 < 0$) is at the right of the circular limit;

between the circular limit and the tangency condition there is a region, of non–negligible size for $a_{target}/a \leq 0$, in which orbits are either totally outside (for $a_{target}/a > -1$) or totally inside (for $a_{target}/a < -1$) that of the target, so that no collision is possible. At the left of the tangency condition there are the crossing, potentially colliding orbits.
Starting from the third panel of Fig. 8, the upper parts of the panels, above the yellow straight line, are similar to the plots of the first two panels of Fig. 8; below that line, the forbidden region is on the left of the circular limit, and the region of non intersecting orbits is at the right of the circular limit, between the latter and the tangency condition.
By means of the iso-probability curves the collision risk associated with each object can be assessed. Moreover, by choosing appropriately the values of the impact velocity U, of the range of masses of the plotted population and of the collision probabilities P_1 and P_2, it is possible to have an immediate quantitative estimate of the level of risk of impacts for selected energy ranges, on the target. Note that either both or just one (or even none) of the iso-probability curve might be displayed in a given plot.
For each plot, the corresponding impact velocity is also written on top of the plot.

3.4 View histograms of the distribution of the collisions

Option 12 allows the analysis of the distribution of the collisional events simulated during one SDM 3.0 run. In particular it can show histograms of the collisions in terms of:

1) Semi-major axis [km] of the target

2) Eccentricity of the target

3) Inclination [degree] of the target

4) Target mass [kg]

5) Projectile mass [kg]

6) Collision velocity [km/sec]

After choosing the type of histogram, the user has to choose the altitude range of interest. Then the user has to select if the collision happened in a given year or in the whole simulation time span have to be considered. Then the user has to choose between histograms related to all the collisional events or to catastrophic collisions only or craterizations only. Finally the number of bins in the histogram have to selected.
Once the choices are entered, the selected plot is shown on the screen and the corresponding postscript file is generated.

4 Long term evolution studies

The long term evolution of the space debris population has been studied with the new SDM 3.0 package. The purpose of the study has been to validate the software, to analyse the effect of a series of mitigation measures recently proposed, and partly also applied, at the international level and to test the sensitivity of the evolution to different models coded inside SDM. Therefore, a number of different simulation scenarios have been devised and simulated, as detailed in the following.

- The REFERENCE case is characterised by a routine launch activity deduced from the traffic observed over the last five years (1999-2003). The input has been adjusted by taking into account the phasing out of obsolete launchers and the introduction of new rocket families, with different hardware and mission characteristics.

  The traffic model consists of three basic components: 56 routine launches per year (including lunar/interplanetary probes) (C.3), 6 servicing missions per year to the International Space Station (ISS), with a planned operational lifetime of 25 years, and the maintenance of 5 constellations (Orbcomm, Iridium, Globalstar, GPS, Glonass), with 1 launch per year per constellation (C.2). Starting in 2005, a new navigation constellation (Galileo) is built and then maintained in Medium Earth Orbit (MEO) (C.2). All constellations have a planned lifetime of 20 years.

  Mission Related Objects (MRO) are released according to the current practices (C.3, C.2) and no de-orbiting or re-orbiting of spacecraft and upper stages is performed at the End-Of-Life (EOL). Concerning on-orbit explosions (C.1), the events considered are based on: the explosion statistics over the last 5 years (1999-2003) , with an average of 2.4 explosions/year; the introduction of mitigation measures on several classes of old and new upper stages systems (e.g., passivation of upper stages after burn); the existence, in orbit, of a large number of old upper stages prone to explode for at least a few decades; the progressive introduction, over the coming 5-25 years, of explosion prevention measures on the systems currently in use, leading to no more explosions after 2030.

  As far as the production of slag is concerned, a minimum use of solid rocket motors is envisaged, based on current and planned practices (the transition to new launchers and larger commercial spacecraft is reducing considerably the reliance on such propulsion systems). Therefore, two solid rocket motor firings are simulated for each GPS mission (perigee and apogee burns) and one of them is considered as well for each Long March 3 GEO injection (C.3). The solid rocket motors used for the GPS perigee manoeuvre and the Long March 3 injection in GEO are discarded after burnout, while the motor used for the GPS apogee manoeuvre is integrated in the satellite structure (C.3, C.2).

- In the MIT_1 case, a first mitigation measure is implemented to the REFERENCE scenario. Namely, no mission related object is released after the year 2020.

- In the MIT_2 scenario, in addition to the mitigation measure of MIT_1, the re-orbiting of spacecraft at EOL is performed. In particular, the GEO satellites are re-orbited at

EOL above GEO, according to the IADC recommendation (Eq. 11). Starting from the year 2010, all the spacecraft with $h_p < 1400$ km, or in high eccentricity orbits crossing the LEO region, are manoeuvred to orbits with a residual lifetime $T_{res} = 75$ years. The spacecraft with perigee height $h_p \geq 1400$ km are re-orbited in a super-LEO storage zone above 2000 km (with a width of 100 km). The adoption of a super-LEO storage zone is dictated by fuel saving considerations and was simulated to allow the comparison of the results with other similar studies carried out worldwide. Nonetheless the hazards related to the adoption of such a mixed policy, due to the accumulation of objects in the storage zone, must be stressed. Hopefully the storage zones could be only a temporary solution, before technological improvements could make the de-orbiting of all the spacecraft viable.

Concerning the upper stages, starting always from the year 2010, all those with perigee height $h_p \geq 1400$ km are left where they are, while those with $h_p < 1400$ km, or in high eccentricity orbits crossing the LEO region, are immediately de-orbited at EOL.

- The MIT_3 scenario repeats the MIT_2 one, with the only difference of $T_{res} = 50$ years.
- The MIT_4 scenario repeats the MIT_2 one, with the only difference of $T_{res} = 25$ years.
- The MIT_5 scenario repeats the MIT_2 one, but in this case $T_{res} = 0$ years, i.e. the spacecraft with $h_p < 1400$ km, or in high eccentricity orbits crossing the LEO region, are immediately de-orbited at EOL.
- The INCREASED scenario repeats the REFERENCE case, with an increment of the *routine* launch activity of 1 % per year.
- The SENS_1 scenario repeats the REFERENCE case with the NASA EVOLVE 4 break-up models for explosions and collisions.
- The SENS_2 scenario repeats the MIT_4 case with the NASA EVOLVE 4 break-up models for explosions and collisions.
- The SENS_3 scenario repeats the MIT_5 case with the NASA EVOLVE 4 break-up models for explosions and collisions.

All the simulations carried out for this study cover a time span of 100 years, from 2004 to 2103. Each scenario has been obtained by averaging 20 independent Monte Carlo runs. The initial debris environment, down to 1 mm, is specified by the ESA MASTER 2001 model.
In the REFERENCE scenario all the objects have been propagated with the DCP ultra-fast propagator. Area-to-mass ratio, low and high intensity explosions, collisions and debris velocity distributions have been represented with the subset of models already described in Section 4 (pp. 40-42) of the SDM 2.0 Final Report [1].

4.1 The mitigation scenarios

Figure 10 shows the evolution of the millimetre sized objects in the whole altitude range, up to 40,000 km. The decrease observed in the first 25 years, during which all the scenarios produce very similar results, is a by-product of two facts: 1) the MASTER 2001 population is dominated, in the size range considered, by the slag produced by the historical solid rocket motor firings; 2) the number of new solid rocket motor firings considered in the future by all the scenarios is quite small, so the rate of slag removal is prevalent with respect to the rate of new slag production.
Another important feature of the evolution presented in Figure 10 is that any scenario presents an absolute minimum around 2025, then results again in a systematic debris growth, apart from the oscillations due to the influence of the solar activity cycle on the atmospheric density. The rebound is due to the small fragments produced by a progressively increasing number of collisions among orbital debris. As expected, such effect is much more pronounced in the INCREASED scenario, with a population rise by more than 50 % after one century. The REFERENCE case results in a final growth of about 15 % with respect to the initial conditions, while the implementation of mitigation measures, as simulated by the scenarios from MIT_1 to MIT_5, may result in a smaller growth, but never lower than approximately 4 %.

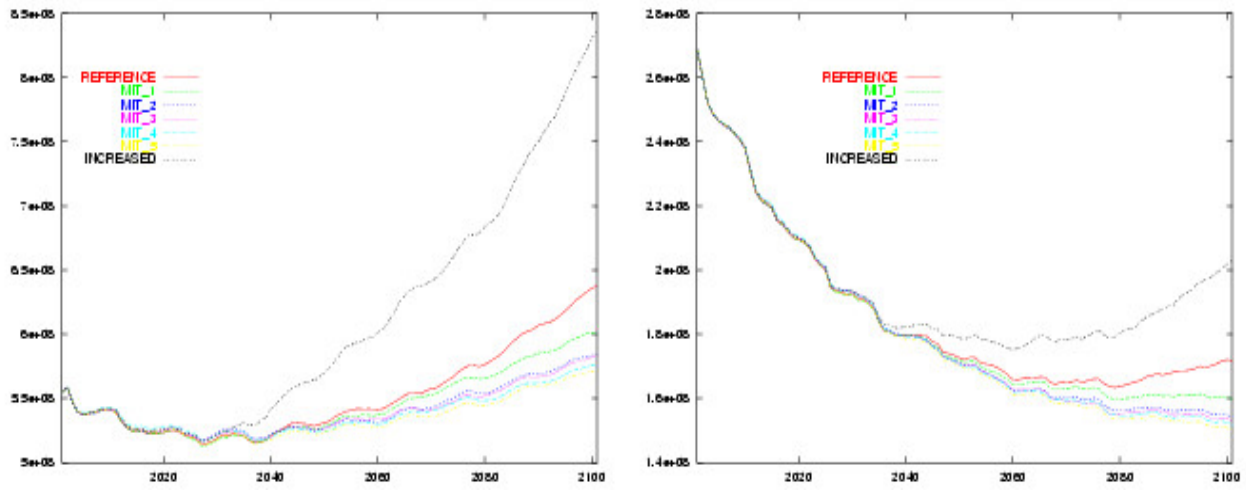

Figure 10: Number of objects with diameter larger than 1 mm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

Focusing the attention below 2000 km, the effect of the slag dominance in the initial population is further exacerbated by the effectiveness of the air drag in removing small particles not replenished at a substantial rate (Figure 10, right panel). All the scenarios display, in fact, a sharp decrease of millimetre sized objects for about 40 years and even the growth of collisional fragments is able to invert the trend only after 80 years in the two non mitigated

cases (REFERENCE and INCREASED). All the mitigated scenarios show either a stabilisation (MIT_1), or a slightly decreasing trend, also in the last 20 years of the time span considered. The final number of particles found after one century is in between 25 % (INCREASED) and 45 % (MIT_5) lower than the initial value. Above 1 cm the MASTER 2001 debris population is dominated by the break-up fragments and the environment evolves in a completely different way. Between 0 and 40,000 km (Figure 12) all the scenarios display very similar results and a moderate growth over the next 40 years. This growth is sustained (in the first 20-30 years) by the remaining on-orbit explosions and by a progressively larger number of collisions. After 2030 collisions remain the only substantial source of centimetre sized objects and a significant increase (more than linear) of the particle number is observed after 2040 for the INCREASED, REFERENCE and MIT_1 scenarios. The increment after one century is about 210 %, 108 % and 70 %, respectively.
Figure 11 shows, on a logarithmic scale, a breakdown of the *running population* (REFERENCE case) for objects larger than 1 cm, according to the different sources. The progressive growth of the collision debris as the dominant source is clearly visible. All the mitigation scenarios

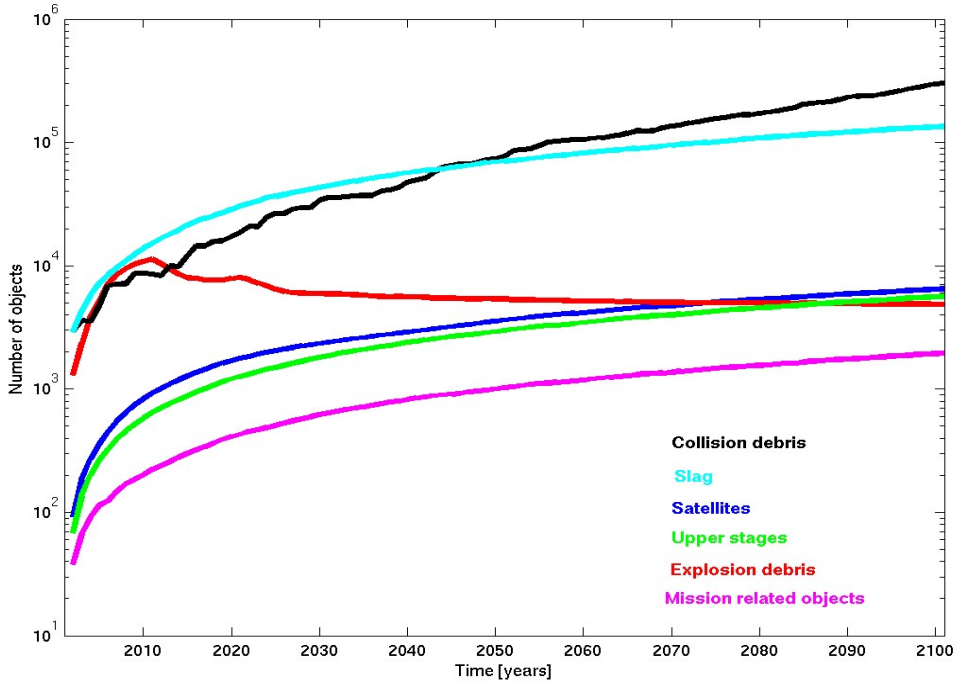

Figure 11: Breakdown, according to the different source mechanisms, of the number of objects in the *running population* with diameter larger than 1 cm in the whole altitude range (REFERENCE case).

envisaging a mix of end-of-life re-orbiting and de-orbiting (from MIT_2 to MIT_5) seems able to maintain approximately stable the growth of centimetre sized debris, at least in the time

span considered, with minor differences between the four cases. However, after 100 years the debris number may rise by an amount in between 36 % (MIT_5) and 50 % (MIT_2). Apart from

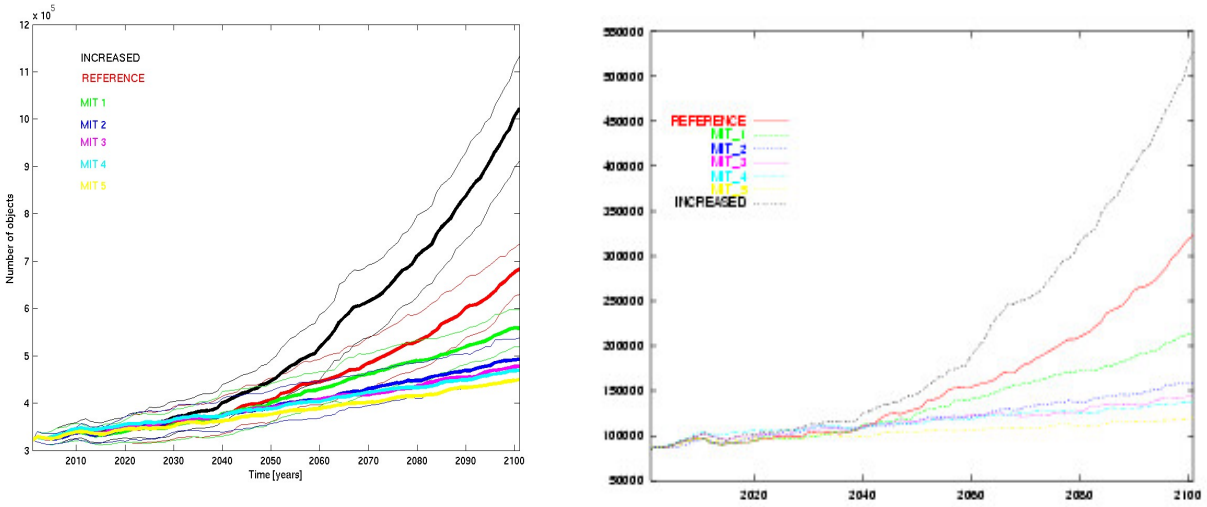

Figure 12: Number of objects with diameter larger than 1 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

the numbers, the same situation applies in LEO, i.e. below 2000 km (Figure 12, right panel), where the air drag plays a very important role. Again, all the simulated cases give quite similar results until 2040, and all the mitigation scenarios envisaging a mix of end-of-life re-orbiting and de-orbiting (from MIT_2 to MIT_5) show a comparable effectiveness (or ineffectiveness) in limiting the growth of centimetre sized debris (by an amount in between 38 % and 83 % after one century).

For objects larger than 10 cm (Fig. 14), the initial growth rate levels off after 2020 for the cessation of most of the on-orbit explosions. In the INCREASED scenario, due to the rising collision probability, the growth rate comes back to values comparable or higher than the initial one after 2040, with an absolute object proliferation of about 250 % in one century. In all other scenarios the growth rate never returns to the values experienced in the first 10-20 years, even though the absolute increase (roughly a doubling in one century) remains significant in the REFERENCE and MIT_1 cases. In the right panel of Figure 14 the 1σ deviation is plotted (thin lines) for the REFERENCE and MIT_2 scenarios to give an idea of the variability of the results with respect to the Monte Carlo average.

Figure 13 shows a breakdown of the *running population* (REFERENCE case) for objects larger than 10 cm, according to the different sources. Note the non-linear growth rate of the collision fragments as opposed to the linear trend of the launch generated objects.

The adoption of a combination of re-orbiting and de-orbiting at the end-of-life is able to drastically reduce the growth of decimetre sized objects above 2000 km, while in LEO the effect is a population stabilisation (with $T_{res} = 75$ or 50 years) or a slightly decreasing trend (with

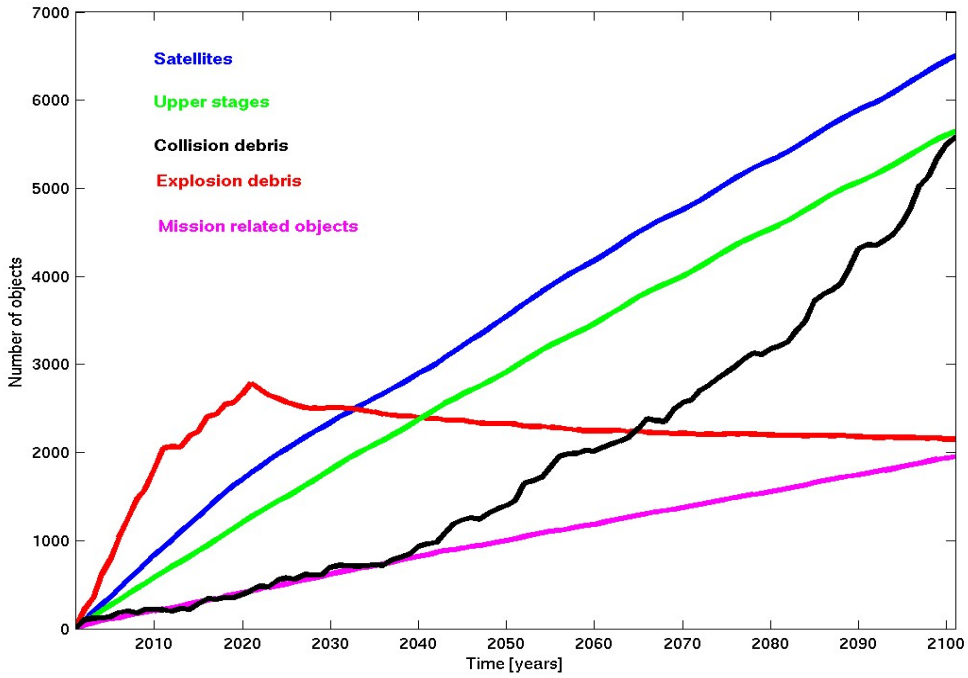

Figure 13: Breakdown, according to the different source mechanisms, of the number of objects in the *running population* with diameter larger than 10 cm in the whole altitude range (REFERENCE case).

T_{res} = 25 or 0 years). The differences among these four scenarios are not substantial: just a reduction of the order of 10 % in the number of objects between the cases with $T_{res} = 75$ and 25, and a further gain of approximately 10 % if an immediate EOL de-orbiting is adopted for any spacecraft with perigee below 1400 km. The population of the objects larger than 1 metre is dominated by the intact spacecraft and upper stages and the trends observed are just the result of the net accumulation due to new launches in the considered time span (Figure 15). As expected, the INCREASED scenario is characterised by a slow exponential growth, with an overall increase of a factor 5 over one century, while the REFERENCE and MIT_1 cases display a linear growth, with a final value more than 3 times larger than the initial one. Even the mitigated cases MIT_2 through MIT_5 attain a linear increase; the slope of the curves is however sensibly smaller than in the REFERENCE and MIT_1 cases. Moreover, it is important to stress that as soon as the lifetime reduction measures reach their full effectiveness against new launches, the constant growth trend becomes the same in all four cases, with a difference of about 20 % in the final number of objects between MIT_2 and MIT_5 (Figure 15). Below 2000 km (Figure 15, right panel) the EOL mitigation measures obtain the important result of stabilising, and then progressively reducing, the number of intact objects in orbit. Even a residual spacecraft lifetime of 75 years in LEO is able to reach this important goal, averting

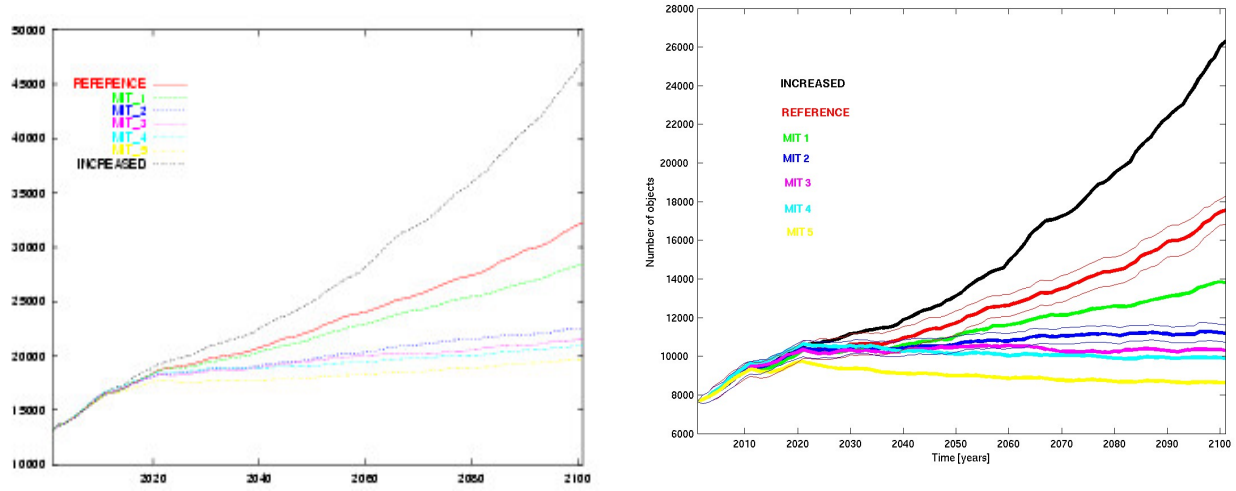

Figure 14: Number of objects with diameter larger than 10 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

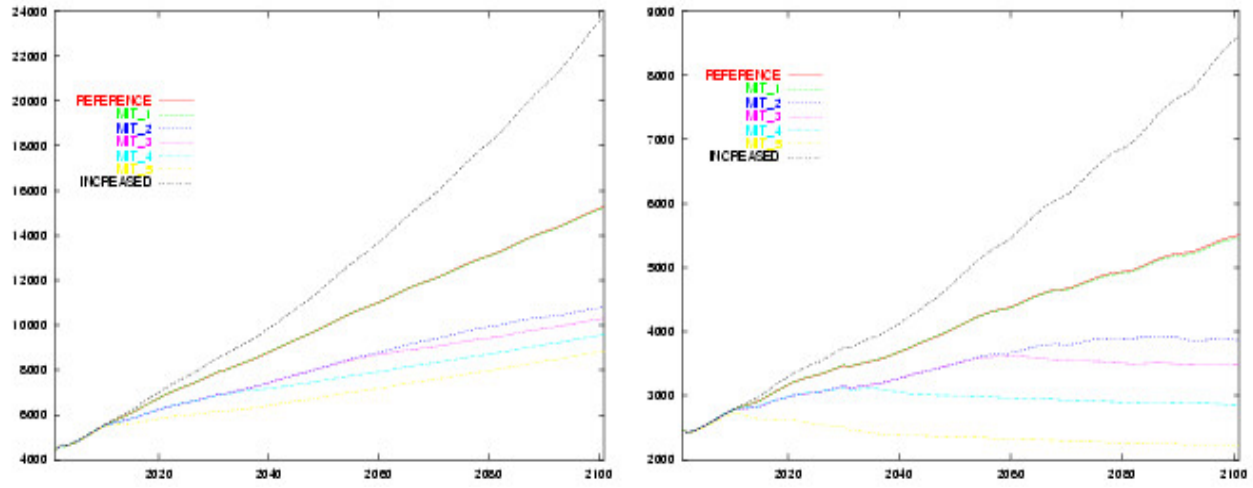

Figure 15: Number of objects with diameter larger than 1 m in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

the long term onset of an exponential growth, but of course there is a temporal delay and a correspondingly higher maximum population with respect to the other three scenarios.

These conclusions are supported by the cumulative number of catastrophic collisions in LEO (Figure 16). In the INCREASED, REFERENCE and MIT_1 scenarios there is a progressive rise of the yearly number of catastrophic collisions, but in the MIT_2 through MIT_5 scenarios the collision rate remains approximately constant, the highest in the MIT_2 case, the lowest in

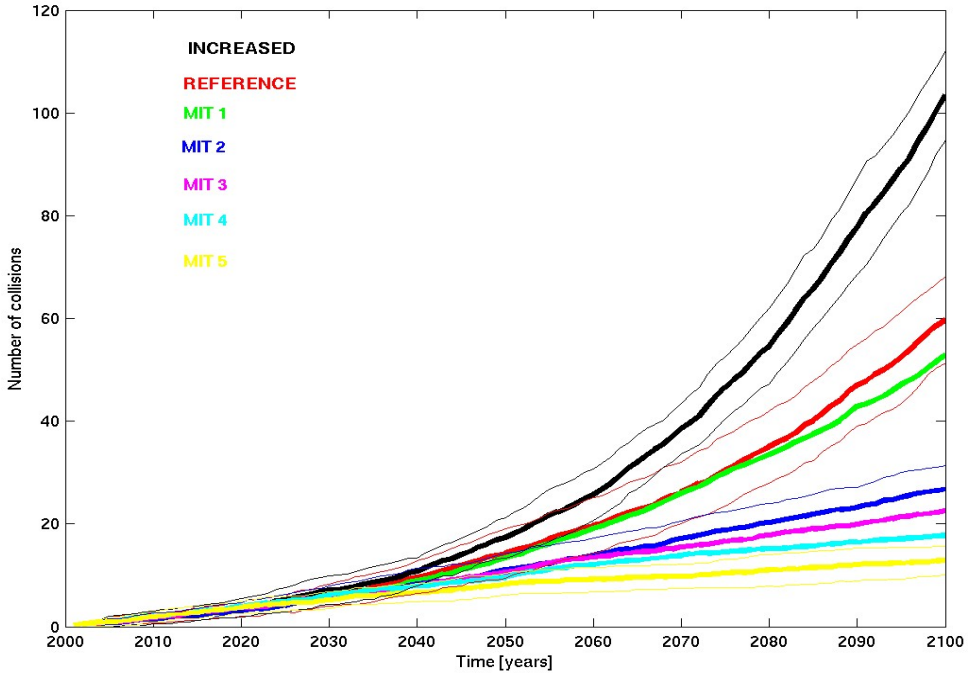

Figure 16: Number of catastrophic collisions in LEO.

the MIT_5 case.
In summary, the MIT_2 through MIT_5 scenarios seem adequately effective in mitigating over the long term the LEO debris environment with the current launch activity.

4.2 The sensitivity analysis

In order to verify the sensitivity of the results as a function of the choice of the models, the REFERENCE case has been simulated also with the new NASA break-up models for explosions and collisions, implemented in SDM 3.0 (SENS_1).

For centimetre sized particles (Figure 17), REFERENCE and SENS_1 give very similar results until 2030, i.e. in the period dominated by the explosions, while there is a progressive divergence afterwards, when collisions become the leading source of debris (see Figure 27). The new NASA models produce a larger number of particles, with a positive difference, after one century, of 12 % in the circumterrestrial space (Figure 17, left panel), and 30 % in LEO (Figure 17, right panel).

Considering decimetre sized objects (Figure 18), the NASA break-up models produce less objects in the explosion dominated era and substantially more objects in the collision dominated era, after 2040-2050 (see Figure 46). This is clearly shown in Figure 19, where the number of collision and explosion fragments larger than 10 cm present in the *running populations* of the REFERENCE and SENS_1 scenarios are compared. At the end of the simulations, the number of objects in SENS_1 is larger by 18 % below 40,000 (Figure 18,left panel) and 33 % (Figure 18, right panel) in LEO.

Even above 1 metre the NASA break-up models produce more collisional fragments (Figure 20). However, the maximum discrepancy with respect to the REFERENCE scenario is less than 15 % in LEO (Figure 20, right panel), after 100 years.

A second sensitivity analysis was carried out by substituting the new NASA break-up models in the MIT_4 scenario (SENS_2). The results for objects larger than 1 cm are presented in Figure 21. In general the agreement between MIT_4 and SENS_2 is quite good in all the time span considered, with discrepancies lower than 10 %.

For diameters greater than 10 cm (Figure 22) the situation is much more complex. During the initial phase, up to 2020, characterised by a substantial amount of explosion fragments, the new NASA break-up models are object-deficient and the numbers in MIT_4 grow much faster. But after the end of the explosions, SENS_2 becomes more and more close to MIT_4, due to a larger number of collision generated fragments. However, in these mitigated cases, in which the target area in orbit remains quite limited in the considered time span, the total number of catastrophic collisions is about 20 (see Figure 27, right panel) and the additional debris produced by the NASA models is not able to close the gap with MIT_4 in just one century.

A qualitative feature of the NASA models results is, in any case, worth to be mentioned: in LEO (Figure 22, right panel) the SENS_2 debris population continues to grow, even if slowly, after one century, while MIT_4 is characterised by a slow population decline. Therefore, if the new NASA models are assumed, the proposed mitigation scenario is not able to stabilise the population of debris larger than 10 cm in LEO.

Above 1 metre the agreement between MIT_4 and SENS_2 is again quite good, both qualitatively and quantitatively, with discrepancies of a few percent (Figure 23).

The third sensitivity analysis was carried out by substituting the new NASA break-up models in the MIT_5 scenario (SENS_3). Even in this case the results for objects larger than 1 cm,

presented in Figure 24, are very similar, irrespective of the break-up models selected for the long term simulations, with differences not exceeding 5 %.
For objects larger than 10 cm (Figure 25) and 1 metre (Figure 26), the behavior of the models and the qualitative properties of the population evolution are the same encountered and discussed in the MIT_4 vs. SENS_2 comparison. In particular, if the new NASA models are assumed, even the immediate EOL removal of spacecraft and upper stages from LEO is not able to stabilise there the population of debris larger than 10 cm. Instead, it starts increasing again after 2060 (Figure 25, right panel), in spite of the modest number of catastrophic collisions (Figure 28), characterised by a slightly decreasing trend.

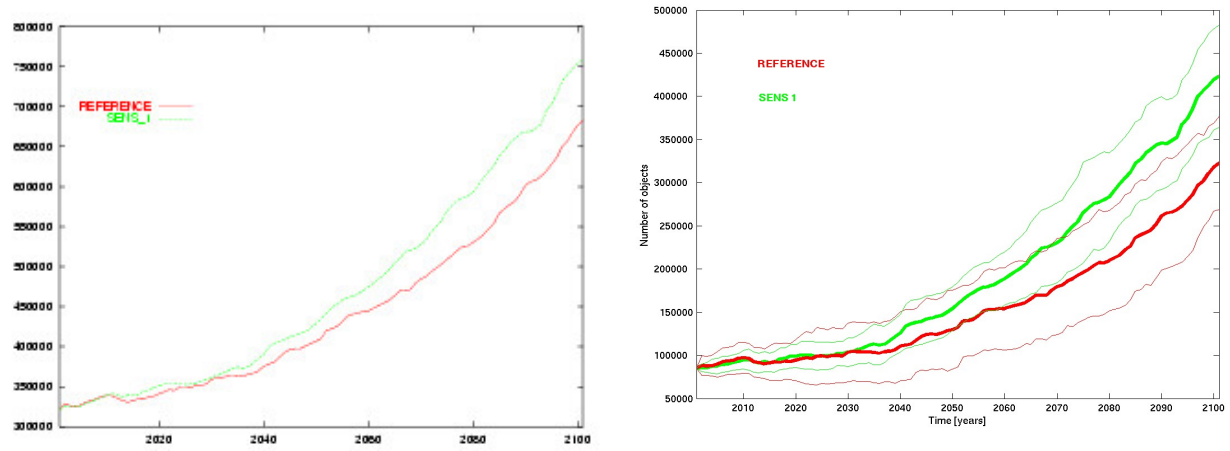

Figure 17: Number of objects with diameter larger than 1 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

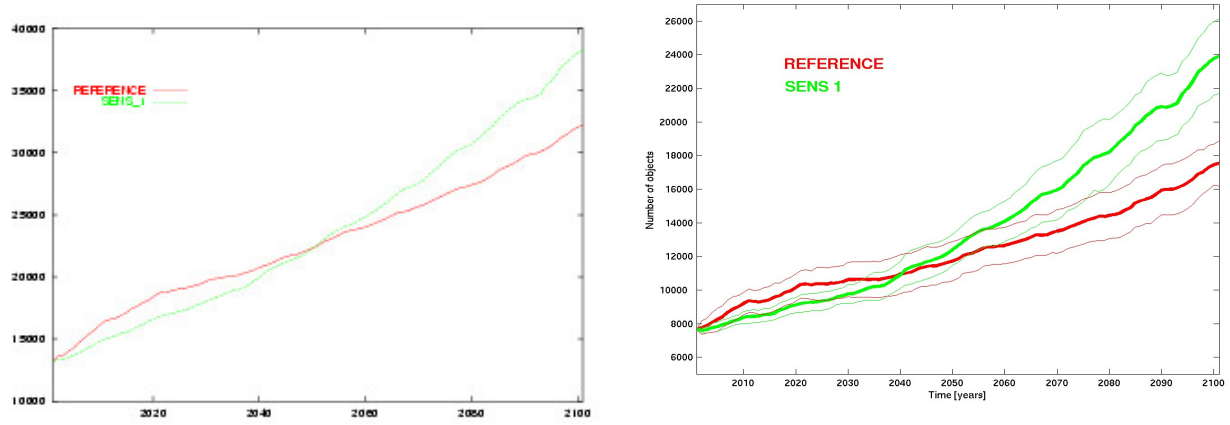

Figure 18: Number of objects with diameter larger than 10 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

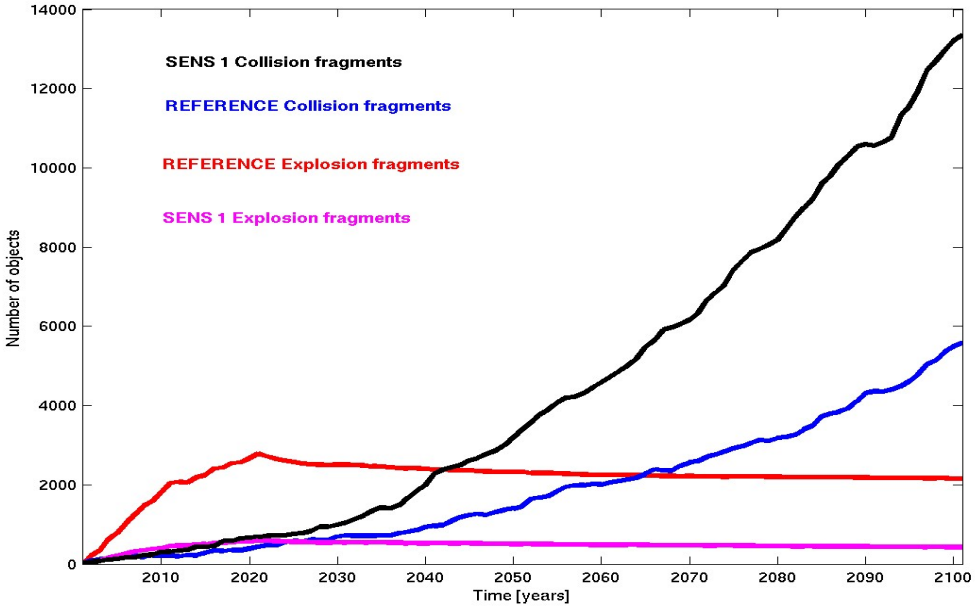

Figure 19: Breakdown of the number of collision and explosion fragments larger than 10 cm, in the whole altitude range, for the REFERENCE and the SENS_1 scenarios.

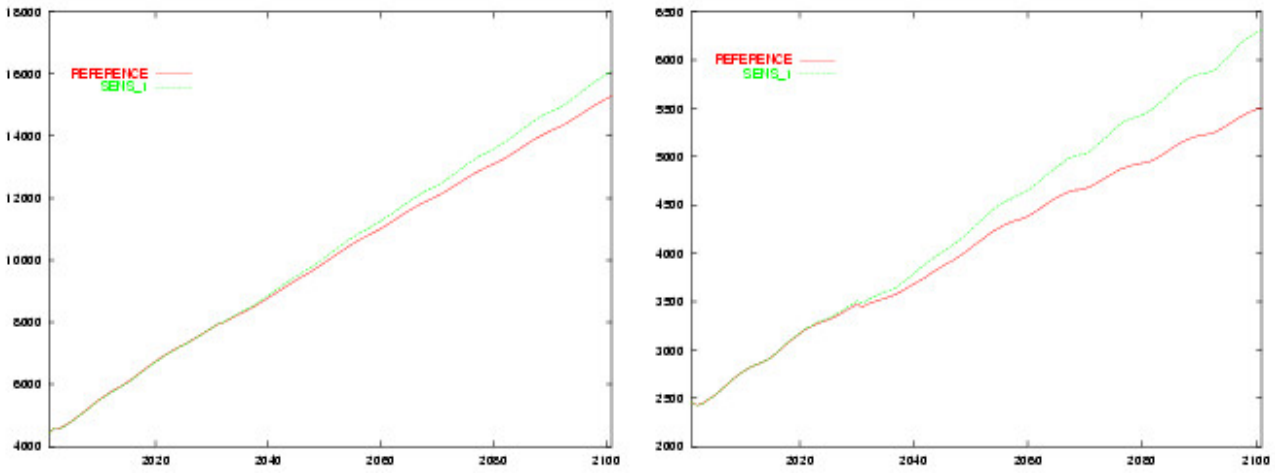

Figure 20: Number of objects with diameter larger than 1 m in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

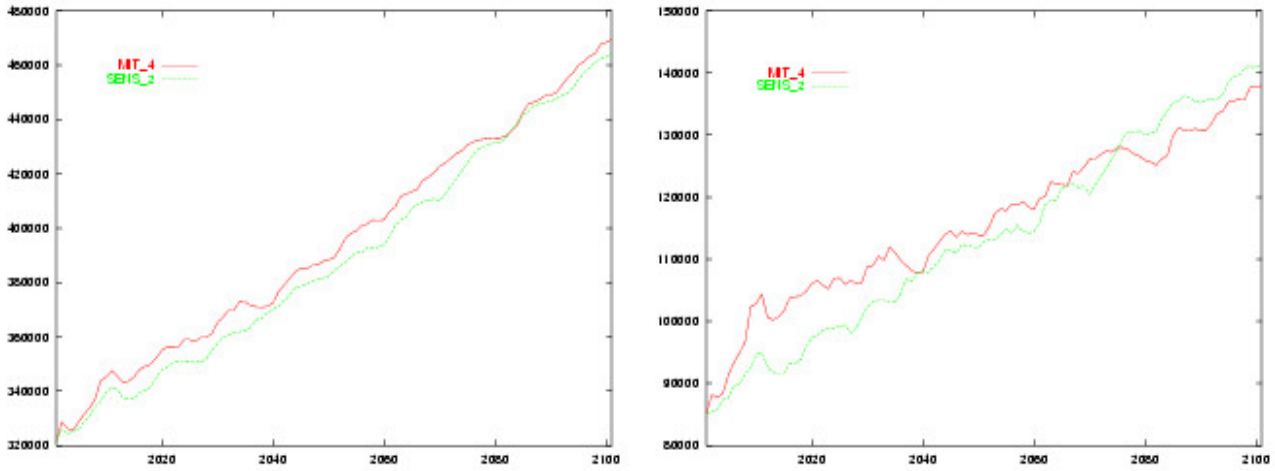

Figure 21: Number of objects with diameter larger than 1 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

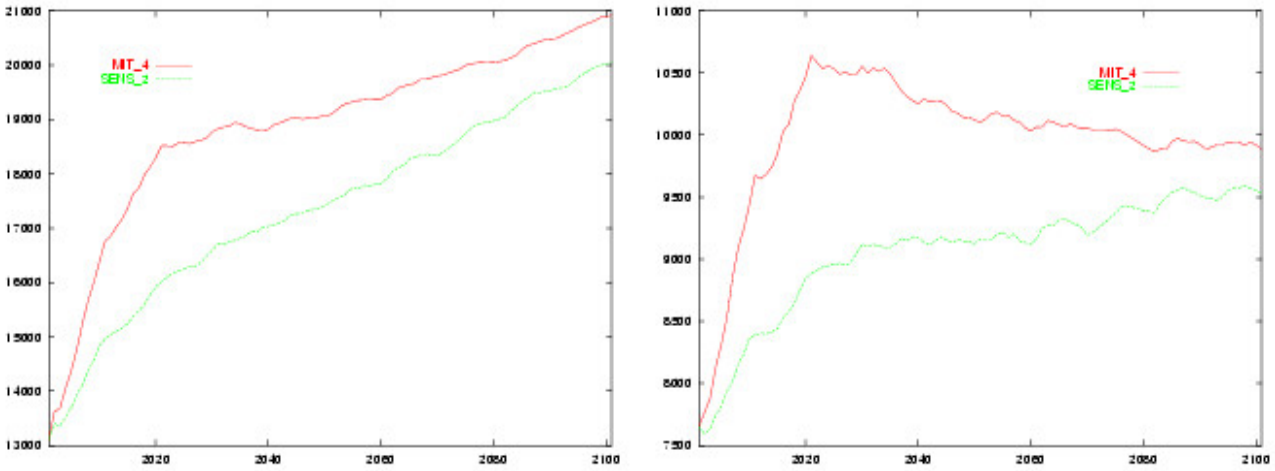

Figure 22: Number of objects with diameter larger than 10 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

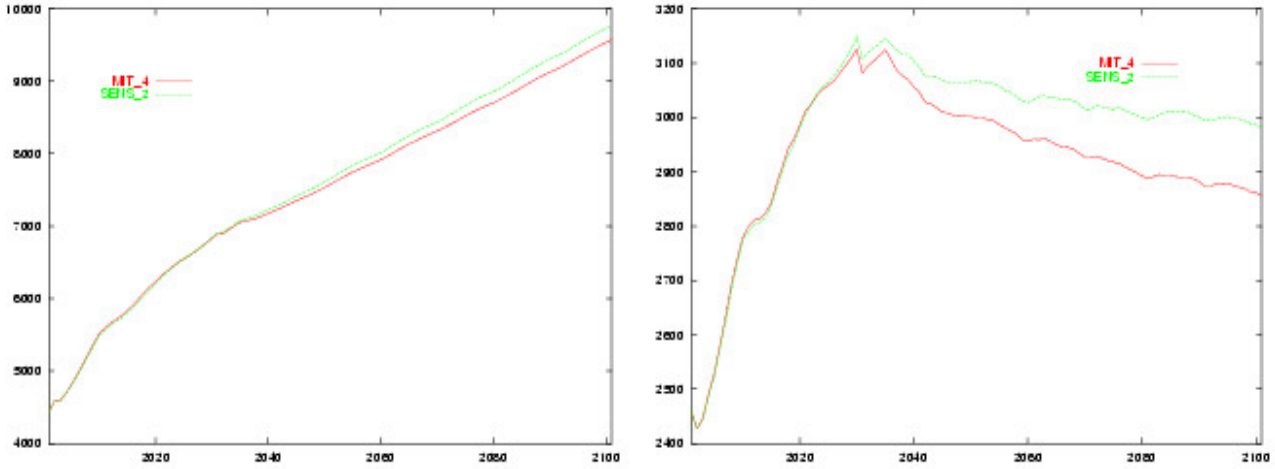

Figure 23: Number of objects with diameter larger than 1 m in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

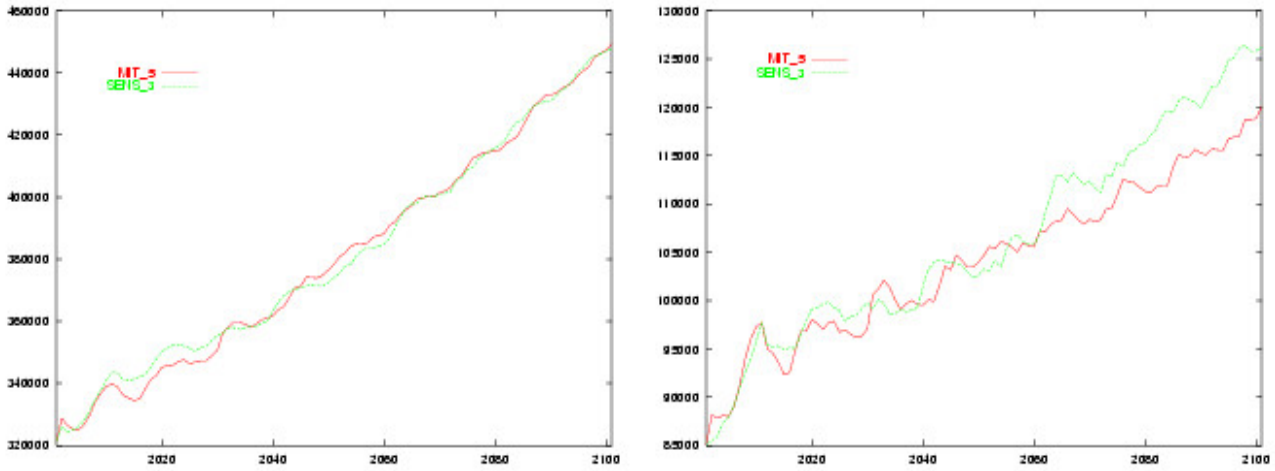

Figure 24: Number of objects with diameter larger than 1 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

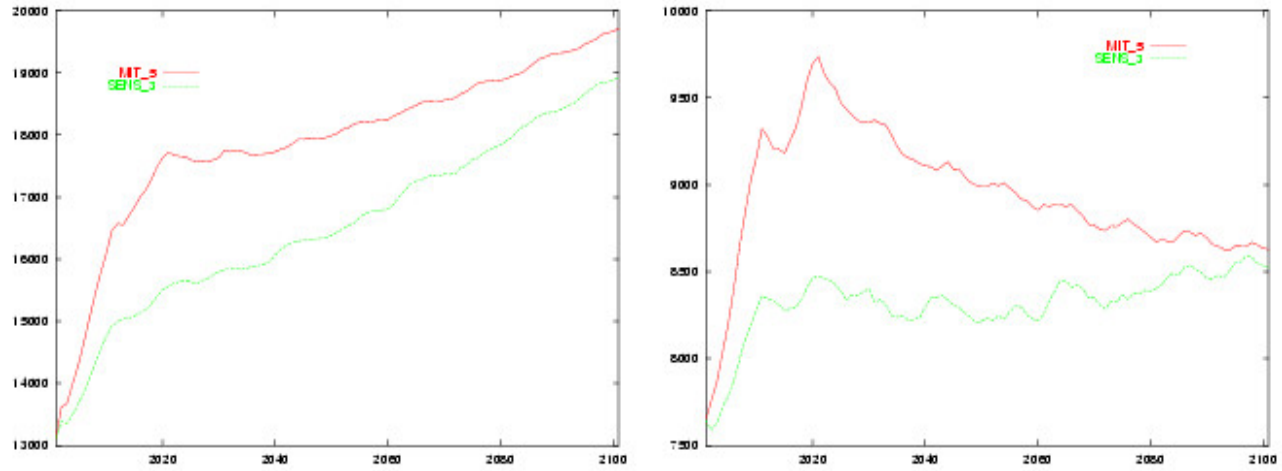

Figure 25: Number of objects with diameter larger than 10 cm in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

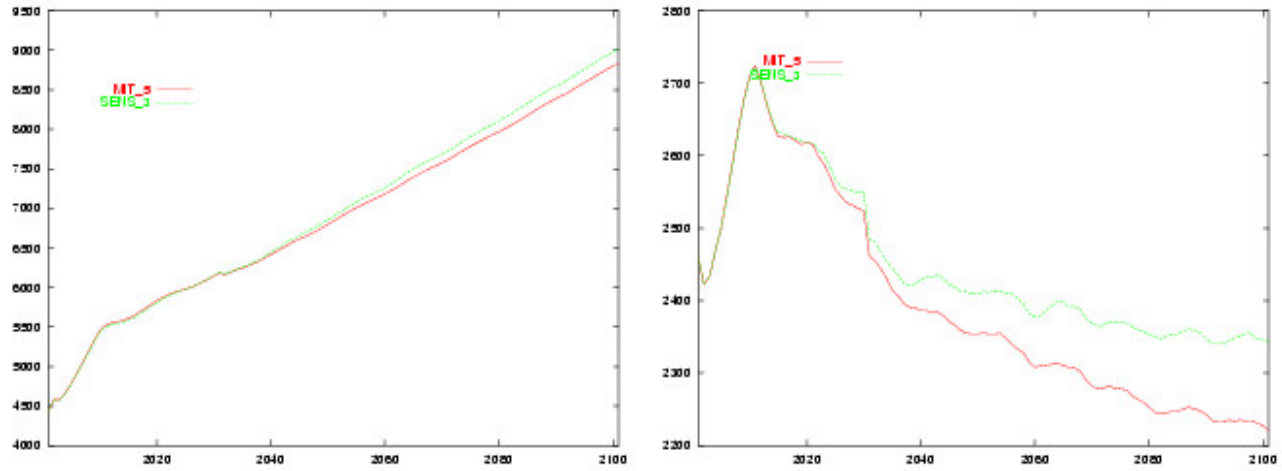

Figure 26: Number of objects with diameter larger than 1 m in the whole altitude range, between 0 and 40 000 km (left panel) and in LEO (right panel).

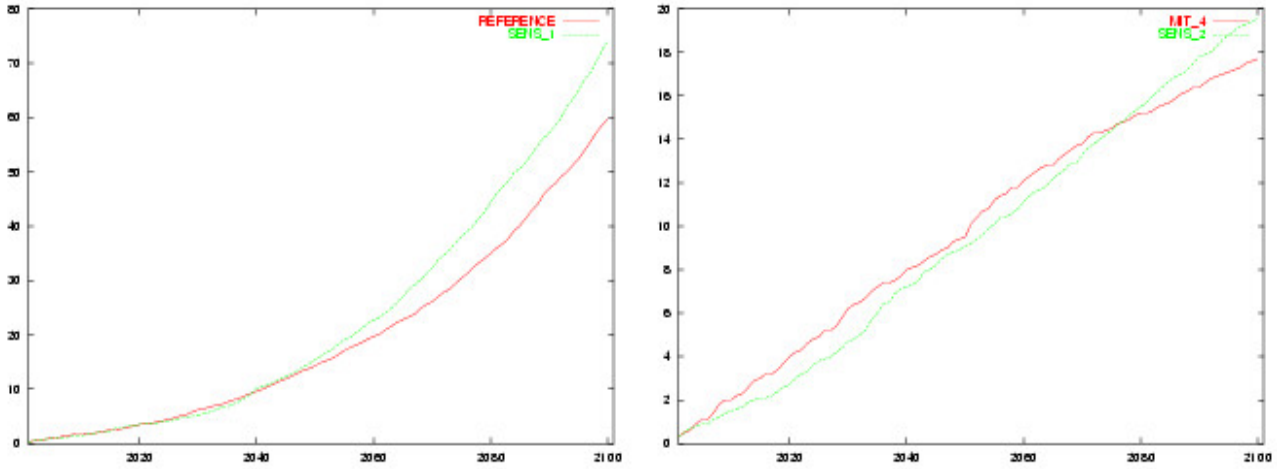

Figure 27: Number of catastrophic collisions in LEO for the first (left panel) and the second (right panel) sensitivity study.

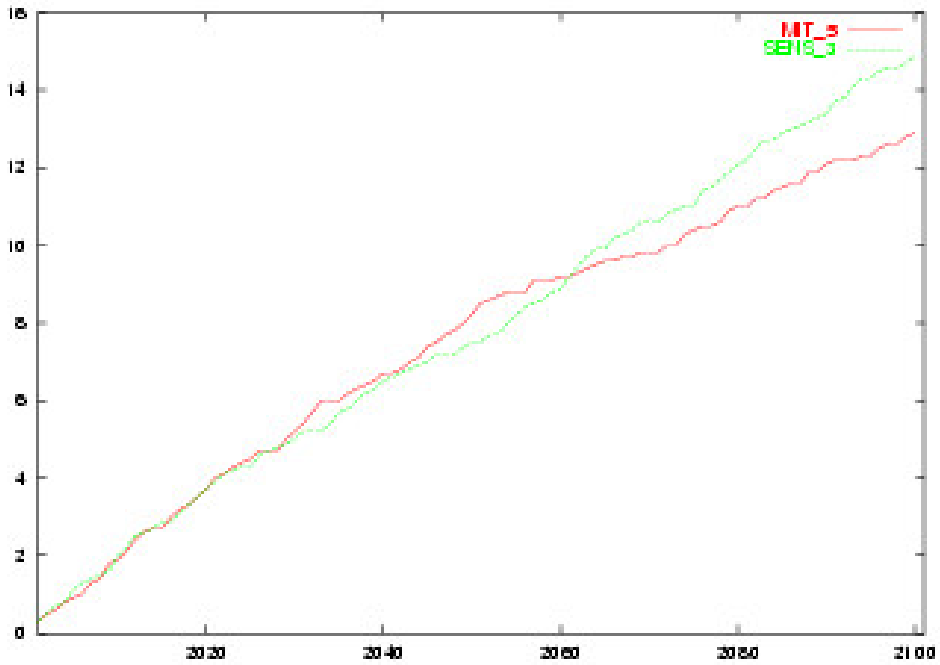

Figure 28: Number of catastrophic collisions in LEO.

5 Summary

The space debris long-term analysis program SDM was developed in the early 90's to study the long-term evolution of orbital debris and to evaluate the effectiveness of mitigation measures. SDM was now upgraded to Version 3.0 including several new features and models for the sources and sinks mechanisms:

- a very accurate propagator including all the relevant gravitational and non gravitational perturbations was added to the existing DCP propagator (Sec. 2.1);
- a model for the production of Solid Rocket Motor (SRM) slag was added (Sec. 2.2);
- the new NASA EVOLVE 4 breakup model to simulate the generation of debris clouds produced by on-orbit explosions and collisions was added (Sec. 2.3);
- a breakup model for low-velocity collisions was added (Sec. 2.4);
- a set of advanced mitigation options for LEO, MEO and GEO were added to allow the simulation of a number of complex scenarios in the different orbital regimes (Sec. 2.5);
- both the pre-processing and the post-processing sections of the code were completely revised and updated (Secs. 1.2 and 3).

Then the long-term evolution of the space debris population with mass larger than 1 mg, from LEO up to 40 000 km of altitude, was studied with the upgraded SDM 3.0 model. The purpose of the study was to test the updated software and to analyse the effect of a series of mitigation measures recently proposed, and partly also applied, at international level and to test the sensitivity of the evolution to different models implemented in SDM.
First a new, up-to-date Business As Usual scenario was outlined, taking into account the changing features of the recent space activities and the most probable future traffic trends. Building on this default scenario, the effectiveness of several mitigation measures was tested, in different orbital regimes. Below 2000 km the envisaged EOL mitigation measures (including passivation and re-orbiting or de-orbiting with different residual lifetimes) obtain the important result of stabilising, and then progressively reducing, the number of objects larger than 10 cm in orbit. Even a residual spacecraft lifetime of 75 years in LEO is able to reach this important goal, averting the long-term onset of an exponential growth, but of course there is a temporal delay and a correspondingly higher maximum population with respect to the other three scenarios.
The sensitivity of the results to the adoption of the new NASA EVOLVE 4 breakup models was then discussed. It was found that, with the adoption of this last breakup model, the population of objects larger than 10 cm in LEO keeps growing, though slowly, even in the mitigated scenarios. Therefore, if it will turn out that the new NASA breakup models are better to simulate future explosions and collisions, the proposed mitigation scenarios are not able to stabilise the population of debris larger than 10 cm in LEO.

A Appendix: software lists

A.1 CHLIFE.F

```
c  ================================================================
c  Version  3.0  of  17/6/2003
c
c  SUBROUTINE CHLIFE
c
c     Check the identity of each object in the running population
c     and adopt the proper END of life disposal (if applicable)
c
c  INPUT ARGUMENTS
c      iear      year of the simulation
c      deflag    de-orbiting on circular (c) or elliptical (e) orbit
c      reslif    residual lifetime of the disposal orbit for LEO
c                satellites [years]
c      au_fi     flag to discriminate between the automatic procedure to
c                de-orbit/re-orbit to the most energy convenient case
c                (au_fi = a)and the procedure which uses the values perdeo
c                to choose the satellites to be re-orbited in the LEO
c                graveyard (au_fi = f)
c      perdeo    perigee altitude of the spacecraft which are re-orbited
c                in a LEO graveyard orbit [km]
c      deoalt    altitude of the LEO circular graveyard orbit [km]
c      nc        vector with the number of objects in each mass bin
c      h1,h2     Amplitudes of the triangular distribution for deorbiting
c      bin       matrix of the running population objects
c
c  OUTPUT ARGUMENTS
c      bin       matrix of the running population objects with the new
c                orbital elements and identification code
c
c   CALL SUBROUTINES
c    deogeo, deorb
c
c   INCLUDE FILES
c    parcol.h, parbin.h, idcode.h
c
c  References
c
c  Analysis
```

```
c    A.Rossi
c
c  Programmer
c    A.Rossi - CNUCE - PISA
c
c  Modifications
c
c  Comments
c
c=================================================================
      SUBROUTINE chlife(iear,deflag,reslif,au_fi,perdeo,deoalt,nc,
     +                  h1,h2, bin)
c
      IMPLICIT none
      INCLUDE 'parcol.h'
      INCLUDE 'parbin.h'
c  INPUT
      INTEGER iear,nc(nbinm)
      DOUBLE PRECISION reslif,perdeo,deoalt
      CHARACTER deflag*1,au_fi*1
      DOUBLE PRECISION h1,h2
c  OUTPUT
      DOUBLE PRECISION bin(npar,nobjx,nbinm)
c
      INTEGER l,i,nn
      DOUBLE PRECISION td
      INCLUDE 'idcode.h'
      INCLUDE 'datdeo.h'
c
c  LINUX
      save
c
      DO l=nfrs,nbinm
         DO 10 i=1,nc(l)
            IF(bin(npar,i,l).lt.idups)THEN
c  Satellites
               IF(bin(npar,i,l).gt.idspa.and.bin(npar,i,l).lt.
     +            idcoco)THEN
                  td=bin(npar,i,l)-idspa
                  IF(td.eq.iear)THEN
                     write(95,94)iear,(bin(nn,i,l),nn=1,npar)
 94                  format(i6,1x,6(f20.7),' BEFORE DEORBITING')
```

```
                    IF(au_fi.eq.'f')THEN
c  use the value of perdeo to choose the satellites to be re-orbited
                       CALL deorb(iear,deflag,reslif,perdeo,deoalt,h1,
     +                             h2, bin(1,i,l))
                    ELSE
                       CALL deorb_au(iear,deflag,reslif,perdeo,deoalt,
     +                                h1,h2, bin(1,i,l))
                    ENDIF
c  check deorbiting
                    bin(6,i,l)=idspa
                    write(95,95)iear,(bin(nn,i,l),nn=1,npar)
 95                 format(i6,1x,6(f20.7),' DEORBITED')
                  ENDIF
                  GOTO 10
               ENDIF
c  Constellation satellites
               IF(bin(npar,i,l).ge.idcoco.and.bin(npar,i,l).lt.
     +            idunco) THEN
                  td=bin(npar,i,l)-idcoco
                  IF(td.eq.iear)THEN
                     write(95,94)iear,(bin(nn,i,l),nn=1,npar)
                     IF(au_fi.eq.'f')THEN
                        CALL deorb(iear,deflag,reslif,perdeo,deoalt,h1,
     +                             h2, bin(1,i,l))
                     ELSE
                        CALL deorb_au(iear,deflag,reslif,perdeo,deoalt,
     +                                h1,h2, bin(1,i,l))
                     ENDIF
                     bin(npar,i,l)=idunco
                     write(95,95)iear,(bin(nn,i,l),nn=1,npar)
                  ENDIF
                  GOTO 10
               ENDIF
c  Controlled and deorbited GEO
               IF(bin(npar,i,l).ge.idcoge.and.bin(npar,i,l).lt.
     +            idndge)THEN
                  td=bin(npar,i,l)-idcoge
                  IF(td.eq.iear)THEN
                     write(95,94)iear,(bin(nn,i,l),nn=1,npar)
                     CALL deogeo(iear,h1,h2,bin(1,i,l))
                     bin(6,i,l)=idunge
                     write(95,95)iear,(bin(nn,i,l),nn=1,npar)
```

```
                  ENDIF
                  GOTO 10
               ENDIF
c  Controlled GEO (but not deorbited)
               IF(bin(npar,i,l).ge.idndge.and.bin(npar,i,l).lt.
     +         idunge)THEN
                  td=bin(npar,i,l)-idndge
                  IF(td.eq.iear)THEN
                     bin(npar,i,l)=idunge
                     write(95,96)iear,(bin(nn,i,l),nn=1,npar)
 96                  format(i6,1x,6(f20.7),' NOT DEORBITED')
                  ENDIF
                  GOTO 10
               ENDIF
            ENDIF
 10      CONTINUE
      ENDDO
      RETURN
      END
c  ===================================================================
c  Version   3.0  of 1/07/2003
c
c  SUBROUTINE DEOGEO
c
c     Routine for the reorbiting of the GEO satellites
c     The satellite semimajor axis is raised by a quantity
c     specified by the IADC formula:
c
c              Delta H = 235 + 1000 * Cr * A/M     [in km]
c
c     The actual semimajor axis is then extracted from a triangular
c     distribution with the Delta H peak value. The eccentricity is
c     brought to the value eccgeo written in    datdeo.h,  while
c     the inclination remains the same.
c
c  INPUT ARGUMENTS
c      iear      current year of the simulation
c      h1,h2     Amplitudes of the triangular distribution for
c                deorbiting
c      vec       vector with the elements of the satellite to
c                be reorbited
c
```

```
c  OUTPUT ARGUMENTS
c      vec       vector with the elements of the reorbited satellite
c
c   CALL SUBROUTINES
c
c   INCLUDE FILES
c    datdeo.h
c
c  References
c
c  Analysis
c    A.Rossi
c
c  Programmer
c    A.Rossi - CNUCE - PISA
c
c  Modifications
c
c  Comments
c
c=====================================================================
      SUBROUTINE deogeo(iear,h1,h2, vec)
c
      IMPLICIT none
      INCLUDE 'parbin.h'
c  INPUT/OUTPUT
      INTEGER iear
      DOUBLE PRECISION h1,h2
      DOUBLE PRECISION vec(npar)
c
      DOUBLE PRECISION dh,tem,ve,dv
      INCLUDE 'datdeo.h'
c
      dh=delt+(dmu*cr*vec(npar_orb+1)/vec(npar_orb+2))
      tem=vec(1)+dh
c  triangular distribution
      CALL trgdeo(tem,h1,h2, ve)
c  calculate necessary Delta V
      CALL del_v_de(2,vec(1),vec(2),ve,eccgeo, dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
           write(86,100)iear,idnint(vec(6)),vec(1),vec(2),ve,eccgeo,dv
```

```
 100        format(i5,1x,i9,2(1x,f10.3,1x,f6.4),1x,f6.3,'  GEO')
      vec(1)=ve
      vec(2)=eccgeo
      RETURN
      END
c  ================================================================
c  Version   3.0  of   13/6/2003
c
c  SUBROUTINE DEORB
c
c     Routine for the deorbiting of the LEO satellites
c
c  INPUT ARGUMENTS
c      iear      current year of the simulation
c      deflag    flag for de-orbiting options
c                  c  = circular orbit
c                  e  = elliptical orbit
c      reslif    residual lifetime of the orbit in which the deorbited
c                satellites are placed
c      perdeo    perigee altitude of the spacecraft which are
c                re-orbited in a LEO graveyard orbit
c      deoalt    altitude of the LEO circular graveyard orbit
c      h1,h2     Amplitudes of the triangular distribution for
c                deorbiting
c      vec       vector with the elements of the satellite to
c                be reorbited
c
c  OUTPUT ARGUMENTS
c      vec       vector with the elements of the reorbited satellite
c
c
c   CALL SUBROUTINES
c    dcp, findal
c
c   INCLUDE FILES
c    datdeo.h, datear.h
c
c  References
c
c  Analysis
c    A.Rossi
c
```

```
c  Programmer
c    A.Rossi - CNUCE - PISA
c
c  Modifications
c
c  Comments
c
c=======================================================================
      SUBROUTINE deorb(iear,deflag,reslif,perdeo,deoalt,h1,h2, vec)
      IMPLICIT none
      INCLUDE 'parbin.h'
c  INPUT
      INTEGER iear
      DOUBLE PRECISION reslif,perdeo,deoalt
      CHARACTER deflag*1
      DOUBLE PRECISION h1,h2
c  OUTPUT
      DOUBLE PRECISION vec(npar)
c
      DOUBLE PRECISION wv(npar-1),perig,erre,apog,tem,ve,veec,dv
      DOUBLE PRECISION veco1,veco2,per,rapo,rperin,adisp,edisp,a,e,di
      DOUBLE PRECISION ar,am,deal,deal0
      INTEGER ica,l,i,nree,ii
c
      INCLUDE 'datdeo.h'
      INCLUDE 'datear.h'
      INCLUDE 'dattri.h'
c
      perig=vec(1)*(1.d0-vec(2))
      apog=vec(1)*(1.d0+vec(2))
c           write(94,94)vec
c 94        format(6f20.11)
c
c  if the satellite is already all above the graveyard, DO nothing
      IF(perig.gt.(deoalt+reart))RETURN
c
c  If the perigee is higher than   perdeo  and lower than  hiper
c  we re-orbit on the LEO graveyard orbit
      IF(perig.ge.(perdeo+reart).and.perig.lt.hiper)THEN
c  if the orbit is circular, all below the LEO graveyard orbit limit
c  we reorbit in a circular orbit in the LEO graveyard zone
         IF(apog.lt.(deoalt+reart))THEN
```

```
            tem=reart+deoalt
c  ----  triangular distribution
            CALL trgdeo(tem,h1,h2, ve)
c calculate necessary Delta v (case 2)
            veec=0.d0
            CALL del_v_de(2,vec(1),vec(2),ve,veec, dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=2
           write(86,100)iear,idnint(vec(npar)),vec(1),vec(2),ve,
     +                  veec,dv,ica
 100        format(i5,1x,i9,2(1x,f10.3,1x,f6.4),1x,f6.3, ' CASE ',i1)
            vec(1)=ve
            vec(2)=0.d0
c           write(94,94)vec
            RETURN
         ELSE
c  if the orbit is elliptical, with perigee below the LEO graveyard
c  orbit limit we reorbit in an elliptical orbit with perigee above
c  the limit and the apogee is not moved
c  ----  triangular distribution
            veco1=vec(1)
            veco2=vec(2)
            CALL trgdeo(deoalt+reart,h1,h2, ve)
            per=ve
            vec(2)=(apog-per)/(apog+per)
            vec(1)=apog/(1.d0+vec(2))
c  calculate necessary Delta v (case 3)
            CALL del_v_de(3,veco1,veco2,vec(1),vec(2), dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=3
            write(86,100)iear,idnint(vec(npar)),veco1,veco2,vec(1),
     $                   vec(2),dv,ica
c           write(94,94)vec
            RETURN
         ENDIF
      ENDIF
c  The de-orbiting is performed only if the satellites
c  in quasi-circular orbits are  within a selected altitude band, while
c  for elliptic orbits the check on lifetime is always performed
c  I.e. if the satellite is in almost circular orbit and has a low
```

```
c  perigee (perig lower than lowper) then it will surely reenter
c  by itself very soon so no forced reentry is performed
      IF(vec(2).le.0.1d0.and.perig.lt.hiper.and.perig.gt.lowper)THEN
c if the residual time is 0 it means that the spacecraft is
c sent directly into the atmosphere
         IF(reslif.eq.0.d0)THEN
c  calculate necessary Delta v (case 1)
c  calculate the elements of the immediate disposal orbit
c  apogee of both the orignal and the disposal orbits
            rapo=vec(1)*(1.d0+vec(2))
c  perigee of the disposal orbit
            rperin=reart+100.d0
c  semimajor axis of the disposal orbit
            adisp=(rapo+rperin)/2.d0
c  eccentricity of the disposal
            edisp=1-(rperin/adisp)
            CALL del_v_de(1,vec(1),vec(2),adisp,edisp, dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=1
            write(86,100)iear,idnint(vec(npar)),vec(1),vec(2),
     $                   adisp,edisp,dv,ica
            vec(1)=0.d0
            vec(2)=0.d0
c           write(94,94)vec
            RETURN
         ENDIF
         DO l=1,npar-1
            wv(l)=vec(l)
         ENDDO
c Check if the reentry happens before reslif years --> no need to deorbit
c A preliminar check is performed if the perigee is higher than a selected
c quantity  perche, taken from datdeo.h
c If this is the case we assume that the deorbiting must always be done
c saving the time to check.
         IF(perig.lt.perche)THEN
            DO  i=1,int(reslif)
               CALL dcp(iear+(i-1),1.d0,1,wv, nree,a,e,di,ar,am)
               IF(nree.eq.0)GOTO 10
c  load working vector used inside this loop
               wv(1)=a
               wv(2)=e
```

```
               wv(3)=di
             ENDDO
           ENDIF
c  If did not reenter within reslif years move the satellite
c  in a circular or elliptic orbit with lifetime of  reslif  years
c  First calculate the altitude of the orbit with the given
c  reslif  years of lifetime, give the area and mass of the satellite
c
c de-orbiting in circular orbit
           IF(deflag.eq.'c')THEN
              CALL findal(reslif,vec, deal)
            deal0=deal
 122        continue
            wv(1)=deal+reart
            wv(2)=0.d0
            DO i=3,npar_orb
               wv(i)=vec(i)
            ENDDO
            wv(npar_orb+1)=vec(npar_orb+1)
            wv(npar_orb+2)=vec(npar_orb+2)
            DO ii=1,100
               CALL dcp(iear+(ii-1),1.d0,1,wv, nree,a,e,di,ar,am)
               IF(nree.eq.0)THEN
                  IF(ii.le.(reslif+3.d0).and.ii.ge.(reslif-3.d0))THEN
c                     write(65,*)iear,' VA BENE, ii = ',ii
                     GOTO 123
                  ELSE IF(ii.lt.(reslif-3.d0))THEN
c  alzo un po' l'orbita e ricontrollo
c                     write(66,*)'DA ',deal,' A ',deal+10.d0
                     deal=deal+10.d0
                     GOTO 122
c       write(66,*)iear,deal,'reentry after ',ii
c  se e' un po' piu' alto non importa
                  ELSE
c                     write(66,*)'TROPPO ALTO'
                     GOTO 123
                  ENDIF
               ENDIF
               wv(1)=a
               wv(2)=e
               wv(3)=di
            ENDDO
```

```
 123      continue
c          write(66,*)iear,deal0,deal
c  since this is an analytical calculation the following istruction
c  is needed to correct a few particular cases
            IF(deal.gt.vec(1)-reart)THEN
               deal=vec(1)-reart
            ENDIF
c  calculate necessary Delta V (case 4)
            veec=0.d0
            CALL del_v_de(4,vec(1),vec(2),deal+reart,veec, dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=4
            write(86,100)iear,idnint(vec(npar)),vec(1),vec(2),
     +                  deal+reart,veec,dv,ica
            vec(1)=deal+reart
            vec(2)=0.d0
c de-orbiting in elliptical orbit
         ELSE IF(deflag.eq.'e')THEN
            veco1=vec(1)
            veco2=vec(2)
            CALL finell(iear,reslif, vec)
c  calculate necessary Delta V (case 1)
            CALL del_v_de(1,veco1,veco2,vec(1),vec(2), dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=1
            write(86,100)iear,idnint(vec(npar)),veco1,veco2,vec(1),
     $                  vec(2),dv,ica
         ELSE
          write(*,'(a)')'Wrong value of the deflag   flag = ',deflag
          write(*,'(a)')'** check the file sdm.dat **'
         ENDIF
c  elliptical orbits
c  In this case we have to analyse all the orbits with perigee
c  inside LEO (but not lower than a given, reasonable value
c  (e.g. lwprel=180 km) since it can have a very low perigee but a high
c  apogee and so stay there for a long time
c
       ELSE IF(vec(2).gt.0.1d0.and.perig.lt.hiper.and.perig.gt.lwprel)
     +then
c if the residual time is 0 it means that the spacecraft is
```

```
c sent directly into the atmosphere
        IF(reslif.eq.0.d0)THEN
c  calculate necessary Delta v (case 1)
c  calculate the elements of the immediate disposal orbit
c  apogee of both the orignal and the disposal orbits
           rapo=vec(1)*(1.d0+vec(2))
c  perigee of the disposal orbit
           rperin=reart+100.d0
c  semimajor axis of the disposal orbit
           adisp=(rapo+rperin)/2.d0
c  eccentricity of the disposal
           edisp=1-(rperin/adisp)
           CALL del_v_de(1,vec(1),vec(2),adisp,edisp, dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
          ica=1
           write(86,100)iear,idnint(vec(npar)),vec(1),vec(2),adisp,
     $                  edisp,dv,ica
           vec(1)=0.d0
           vec(2)=0.d0
c          write(94,94)vec
           RETURN
        ENDIF
        DO l=1,npar-1
           wv(l)=vec(l)
        ENDDO
c  Check if the spacecraft reenters by itself before reslif years
        IF(perig.lt.perche)THEN
           DO i=1,int(reslif)
              CALL dcp(iear+(i-1),1.d0,1,wv, nree,a,e,di,ar,am)
              IF(nree.eq.0)GOTO 10
c  load working vector used inside this loop
              wv(1)=a
              wv(2)=e
              wv(3)=di
           ENDDO
        ENDIF
c  If did not reenter within reslif years move the satellite
c  in a circular or elliptic orbit with lifetime of  reslif  years
c  First calculate the altitude of the orbit with the given
c  reslif  years of lifetime, give the area and mass of the satellite
c
```

```
c de-orbiting in circular orbit
         IF(deflag.eq.'c')THEN
            CALL findal(reslif,vec, deal)
c  since this is an analytical calculation the following istruction
c  is needed to correct a few particular cases
            IF(deal.gt.vec(1)-reart)THEN
               deal=vec(1)-reart
            ENDIF
c  calculate necessary Delta V (case 4)
            veec=0.d0
            CALL del_v_de(4,vec(1),vec(2),deal+reart,veec, dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=4
            write(86,100)iear,idnint(vec(npar)),vec(1),vec(2),
     +                  deal+reart,veec,dv,ica
            vec(1)=deal+reart
            vec(2)=0.d0
c de-orbiting in elliptical orbit
         ELSE IF(deflag.eq.'e')THEN
            veco1=vec(1)
            veco2=vec(2)
            CALL finell(iear,reslif, vec)
c  calculate necessary Delta V (case 1)
            CALL del_v_de(1,veco1,veco2,vec(1),vec(2), dv)
c  write year, type of object, initial semimajor axis and eccentricity,
c  final semimajor axis and eccentricity, delta V
            ica=1
            write(86,100)iear,idnint(vec(npar)),veco1,veco2,vec(1),
     $                  vec(2),dv,ica
         ELSE
          write(*,'(a)')'Wrong value of the deflag   flag = ',deflag
          write(*,'(a)')'** check the file sdm.dat **'
         ENDIF
       ENDIF
 10    continue
c           write(94,94)vec
       RETURN
       END
c  ===============================================================
c
c  Version   3.0  of 17/6/2002
```

```
c
c  SUBROUTINE FINDAL
c
c     Find the altitude of the circular orbit of given lifetime  reslif
c     for a satellite with a given area/mass ratio; uses Eq. (64) of
c     the Study Note of WP 2300 of ESOC Contract No. 10034/92/D/IM(SC)
c
c  INPUT ARGUMENTS
c      reslif    residual lifetime of the disposal orbit for LEO
c                satellites [years]
c      vec       vector with the elements of the satellite to be reorbited
c
c  OUTPUT ARGUMENTS
c      deal      altitude of the circular disposal orbit
c
c   CALL SUBROUTINES
c     solflux, td88m
c
c   INCLUDE FILES
c     datdcp.h, datdeo.h
c
c  References
c     Study Note of WP 2300 of ESOC Contract No. 10034/92/D/IM(SC)
c
c  Analysis
c    A.Rossi
c
c  Programmer
c    A.Rossi - CNUCE - PISA
c
c  Modifications
c
c  Comments
c
c=====================================================================
      SUBROUTINE findal(reslif,vec, deal)
c
      IMPLICIT none
c  INPUT
      DOUBLE PRECISION reslif,vec(6)
c  OUTPUT
      DOUBLE PRECISION deal
```

```
c
      DOUBLE PRECISION res,asm,rap,fa,refro,refsch
      INCLUDE 'datdcp.h'
      INCLUDE 'datdeo.h'
c  Find density and scale height at the reference altitude in the
c  year iear
      CALL td88m(refflu,refh, refro,refsch)
      res=reslif*365.25d0
      asm=vec(4)/vec(5)
      rap=refro*1d-9/refsch
      fa=dlog(1.e13*res*asm*rap)
      deal=refh+fa*refsch
      RETURN
      END
c  ===================================================================
c
c  Version  3.0  of  17/6/2002
c
c  SUBROUTINE FINELL
c
c     Find the perigee of the elliptical disposal orbit with
c     a residual lifetime of   reslif  years
c     for a given satellite
c
c  INPUT ARGUMENTS
c      iear      current year of the simulation
c      reslif    residual lifetime of the disposal orbit for LEO
c                satellites [years]
c      vec       vector with the elements of the satellite to be
c                re-orbited
c
c  OUTPUT ARGUMENTS
c      vec       vector with the new elements of the re-orbited
c                satellite
c
c   CALL SUBROUTINES
c     dcp
c
c   INCLUDE FILES
c     datdcp.h, datdeo.h
c
c  References
```

```
c
c  Analysis
c    A.Rossi
c
c  Programmer
c    A.Rossi - CNUCE - PISA
c
c  Modifications
c
c  Comments
c
c=======================================================================
      SUBROUTINE finell(iear,reslif,vec)
      IMPLICIT none
      INCLUDE 'parbin.h'
c  INPUT
      INTEGER iear
      DOUBLE PRECISION reslif
c  INPUT/OUTPUT
      DOUBLE PRECISION vec(npar)
c
      DOUBLE PRECISION wv(npar-1),wv0(npar-1),deel,peri,dnewpe0,dnewpe1
      DOUBLE PRECISION apo,dnewper,a,e,di,ar,am
      INTEGER loop,m,i,nree,l
c
      INCLUDE 'datdcp.h'
      INCLUDE 'datdeo.h'
      INCLUDE 'datear.h'
      INCLUDE 'dattri.h'
c
      loop=0
      deel=deelpe+reart
c  Deorbiting on elliptical orbit
      peri=vec(1)*(1.d0-vec(2))
      dnewpe0=peri
      dnewpe1=deel
      apo= vec(1)*(1.d0+vec(2))
c  initial value of the perigee for the bisection loop
      dnewper=deel+(peri-deel)/2.d0
c  begin bisection loop
 10   IF(loop.eq.20)THEN
         write(*,*)'Bisection loop for de-orbiting exceeding  20'
```

```
         write(*,*)'Problems in SUBROUTINE finell'
         write(*,*)vec
         stop
      ENDIF
      loop=loop+1
      wv(2)=(apo-dnewper)/(apo+dnewper)
      wv(1)=apo/(1.d0+wv(2))
      DO i=3,npar_orb
         wv(i)=vec(i)
      ENDDO
      wv(npar_orb+1)=vec(npar_orb+1)
      wv(npar_orb+2)=vec(npar_orb+2)
      DO m=1,npar-1
         wv0(m)=wv(m)
      ENDDO
c
      DO i=1,int(reslif)
c        DO m=1,5
c           wv0(m)=wv(m)
c        ENDDO
         CALL dcp(iear+(i-1),1.d0,1,wv, nree,a,e,di,ar,am)
         wv(1)=a
         wv(2)=e
         wv(3)=di
         wv(npar_orb+1)=ar
         wv(npar_orb+2)=am
         IF(nree.eq.0.and.i.eq.int(reslif))THEN
            DO l=1,npar-1
               vec(l)=wv0(l)
            ENDDO
            RETURN
         ELSE IF(nree.eq.0.and.i.lt.int(reslif))THEN
c  if the spacecraft re-enters before   reslif   years
c  we lowered too much the perigee
c  new bisection and back to the orbit propagation
            dnewpe1=dnewper
            dnewper=dnewper+(dnewpe0-dnewper)/2.d0
            GOTO 10
         ENDIF
      ENDDO
c  if the spacecraft did not reenter in the reslif time interval
c  we have to further lower the perigee
```

```
      dnewpe0=dnewper
      dnewper=dnewpe1+(dnewper-dnewpe1)/2.d0
      GOTO 10
      RETURN
      END
c  ============================================================
c
c  Version  3.0  of  2/7/2003
c
c  SUBROUTINE TRGDEO
c
c     Extracts a value from a triangular distribution having peak
c     value h0 and a width of - h1  and  +h2 (taken from the
c     common datdeo.h)
c
c  INPUT ARGUMENTS
c      h0        peak value of the triangular distributon
c      h1,h2     Amplitudes of the triangular distribution
c                for deorbiting
c
c  OUTPUT ARGUMENTS
c      x         value extracted from the triangular ditribution
c
c   CALL SUBROUTINES
c      ran2
c
c   INCLUDE FILES
c      datdeo.h
c
c  References
c
c  Analysis
c    A. Cordelli - CPR - Pisa
c
c  Programmer
c    A.Rossi - CNUCE - PISA
c
c  Modifications
c
c  Comments
c
c============================================================
```

```
      SUBROUTINE trgdeo(h0,h1,h2, x)
      IMPLICIT none
c  INPUT
      DOUBLE PRECISION h0,h1,h2
c  OUTPUT
      DOUBLE PRECISION x
c
      DOUBLE PRECISION dh1,dh2,fac,r,ran2
      INTEGER idum
c
      INCLUDE 'datdeo.h'
      COMMON/rando/idum
c lower limit of the triangular distribution
      dh1=h0-h1
c  upper limit of the triangular distribution
      dh2=h0+h2
      fac=h1/(h2+h1)
      r=ran2(idum)
      IF(r.lt.fac)THEN
         x=dh1+dsqrt(r*(h2+h1)*h1)
      ELSE
         x=dh2-dsqrt((1.d0-r)*(h2+h1)*h2)
      ENDIF
      RETURN
      END
```

A.2 COLGEN.F

```
c  ============================================================
C   Version   3.0  of 28/5/02
C   PURPOSE
C    The Collision Manager  colgen   manages the collisions
c    routines. That is, it reads the file with the collision
c    velocities, calculates the probabilities of
c    collision, then calculates if a collision occurs, generates
c    the fragments and calculates the orbital elements of each fragment
c
C
c  input parameters
c      iear     =  year
c      ilast    =  flag set to 1 if the routine is called one last
c                  time just to write in output the flux and the
```

```
c                  collision probability
c      dt       =  time step (i.e. time interval in which the
c                  routines calculates if a collision has occurred)
c      dens     =  density of the catalogued and "historical" objects
c                  (obj/km^3)
c      xmi      =  lower limits of mass bins
c      xmf      =  upper limits of mass bins
c      c3       =  velocity models flag (h=high, l=low, i=intermediate)
c      c6       =  flag to choose the mass distribution for collisions
c      hanada   = flag for the use of Hanada's model for low velocity
c                   impacts
c                 = 1   use Hanada's model if below limit velocity
c                       (in hanada.h)
c                 = 0   do not use Hanada's model even if below limit
c                   velocity (c=CNUCE, e = EVOLVE)
c      dnoc     =  matrix with the density of the active constellation
c                  and GEO satellites to be subtracted from the actual
c                  density in the calculation of the collision
c                  probability
c      ammin    =  the treshold value under which masses are considered
c                  not one by one but as a distribution (Kg);
c      ammin0   =  lowest mass considered [kg]
c
c  output parameters
C      ntot0    =  total number of fragments generated by all
c                  the collisions in the interval dt
c      binc     =  matrix with the objects
c      nca      =  vector with the number of objects for each mass bin
c
C   CALL SUBROUTINES
C      zero, xtract, xtrobj, cldgen, dnscld, cloud, makbin,
c      gensma, impila
C
c   INCLUDE FILES
c    parexp.h, parcol.h, parbin.h, paralt.h
c    dattri.h, datbin.h, datear.h, datk.h, idcode.h
c
c  References
c    Study Note of WP 2300
c
c  Analysis
c    L.Anselmo, A.Cordelli, P.Farinella, A.Rossi
```

```
c  Programmer
c    A. Rossi, A.Cordelli, L.Anselmo - CPR - PISA
c
c  Modifications
c
c    Select the fragmentation threshold (ss) between two values.
c
c    The controlled constellations satellites are subtracted from the
c    projectile density in the calculation of the collision
c    probability (so they do not impact against other constellation
c    satellites). (22/9/1997)
c
c  Comments
c
c=======================================================================
      SUBROUTINE colgen(iear,ilast,dt,dens,xmi,xmf,c3,c6,hanada,dnoc,
     $                  ammin,ammin0, ntot0,binc,nca)
      IMPLICIT none
      INCLUDE "parexp.h"
      INCLUDE "parcol.h"
      INCLUDE "parbin.h"
      INCLUDE "paralt.h"
c  Number of sampled bins
      INCLUDE 'combin.h'
c  INPUT
      INTEGER iear,ilast,ntot0
      DOUBLE PRECISION dt,dens(nhshx,nbinm),xmi(nbinm),xmf(nbinm)
      DOUBLE PRECISION dnoc(nhshx,nbinm),ammin,ammin0
      CHARACTER*1 c3,c6
c  OUTPUT
      DOUBLE PRECISION binc(npar,nobjbx,nbinm)
      INTEGER nca(nbinm)
c
      DOUBLE PRECISION p(nbin),colbjw(npar+1,nobjx),elem(6)
      DOUBLE PRECISION vx(nobjx),vy(nobjx),vz(nobjx),small(nsmall)
      DOUBLE PRECISION colrat(nbinm,nhshx),procol(nbinm,nhshx)
      DOUBLE PRECISION vcol(nhshx),sobj(npar+1),dnel(nhshx),eleli(5)
      DOUBLE PRECISION biw(npar,nsamx,nsmall),bicw(npar,nobjbx,nbinm)
      DOUBLE PRECISION am(nobjx),ar(nobjx),densac(nhshx,nbinm)
      DOUBLE PRECISION elim(nhshx,nbinm),delta,vol(nhshx)
      DOUBLE PRECISION ddmm,ss1,ss2,hgh,radi,xm1,dam1,a_m,d1,xm2,dam2
      DOUBLE PRECISION d2,are,prob,amp,amt,dn,ran2,ss,di,vcolli,xm1g
```

```
      DOUBLE PRECISION e,a,rano,arg,anom,bb,bk,hcen,rgauss,vlinf
      INTEGER ncaw(nbinm),nmm(nbinm),ia_m,idum,i,nm,nb1,nb2,ind,indp
      INTEGER indt,n0,ntot,indel,n,nojs,l,ibm,noj,nexo,nexo1,lem,ll
      INTEGER npro,ii,kron,hanada
      LOGICAL first
c  EVOLVE 4 model
      DOUBLE PRECISION se
      INTEGER ev_exp,ncaevo(nobjx),nlarg,num_sma,mi,im,mm
      CHARACTER sc_rb*1
c
      CHARACTER c2*1,c5*1,nmom*1,tipo*7
c
      COMMON/elimin/elim
      COMMON/areama/ia_m
      COMMON/volume/delta,vol
      COMMON/rando/idum
      INCLUDE "dattri.h"
      INCLUDE "datbin.h"
      INCLUDE "datear.h"
      INCLUDE "altdat.h"
c multiplicative factor for the impact velocity in the collision
c probability calculations
      INCLUDE "datk.h"
      INCLUDE "idcode.h"
c  Average masses of the mass bins in MASTER for the collision rate
c  calculation
      INCLUDE "colobj.h"
c
      DATA first/.true./
c  LINUX VERSION
      SAVE
c
c logaritmic amplitude of the mass bin
      ddmm=10.d0
      IF(first)THEN
c  write in a file the elements of the colliding object
         WRITE(54,154)
 154   FORMAT('#Year    a        e       i     Mass T.        Mass P.',
     +'     Type     Num. total   Num. objects  V. Coll')
         WRITE(54,155)
 155   FORMAT('#                                                    ',
     +'                          propagated    [km/sec]')
```

```
         WRITE(55,50)
 50      FORMAT('# Flux for each mass bin (m**(-2) yr**(-1))')
         WRITE(56,156)
 156     FORMAT('# Collision probability for each mass bin')
         first=.false.
c  Read the file with the treshold between fragmentation and
c  craterization (in joule/kg)
c  read central altitude of the shell and collision velocity for
c  the shell (km/sec)
         READ(12,*)
         READ(12,*)
c  CHANGE: READ UPPER AND LOWER LIMIT OF THRESHOLD
         READ(12,*)ss1,ss2
c  800 shells of 50 km heigth
         DO i=1,nhshx
            READ(12,*,end=20)hgh,vcol(i)
         ENDDO
 20      CONTINUE
         close(12)
      ENDIF
      ntot0=0
      CALL zero(nmm,nbinm)
      CALL zero(nca,nbinm)
      CALL zero(ncaw,nbinm)
c  radius of the altitude bins
      radi=(rmax-rmin)/dfloat(nhshx)
c  zero the procol matrix with the collision probability
      CALL zeroma(procol,nbinm,nhshx)
c  subtract the density of the
c  controlled constellation and GEO satellites (which are supposed NOT to
c  collide with the other satellites)
      CALL difmat(dens,dnoc,nhshx,nbinm, densac)
c  loop 31 on altitude shells
      DO 31 nm=1,nhshx
c  center of the altitude shell
         hcen=delta*nm-delta/2.d0
c  loop 30 on mass bins for the target
         DO 30 nb1=1,nbinm
c  Get the mass of the target from the average values in the
c  colobj.h file. A different mass for LEO satellites (shell below
c  10000 km), MEO (shells between 10000 and 40000 km) and GEO
            IF(hcen.LE.1.d4)THEN
```

```
               xm1=av_leo(nb1)
c               CALL lognor(dlog(av_leo(nb1)),std_leo(nb1), xm1g)
c               xm1g=rgauss(av_leo(nb1),std_leo(nb1))
c               if(nb1.eq.10)  write(59,*)xm1,xm1g
            ELSE IF(hcen.GT.1.d4.AND.hcen.LE.4.d4)THEN
               xm1=av_meo(nb1)
            ELSE
               xm1=av_geo(nb1)
            ENDIF
c  get the area from the mass
            dam1=a_m(xm1,ia_m)
c  get the diameter from the area
            d1=dsqrt(dam1/pi)
c  flux of particles  of the mass bin nb1 at altitude shell  nm
c  (in m**-2 year**-1)
c  (31.5 transforms from km**-2 sec**-1 to m**-2 year**-1)
               colrat(nb1,nm)=dens(nm,nb1)*alfa*vcol(nm)*31.5d0
c  loop 32 on mass bins for the projectile
            DO 32 nb2=nb1,nbinm
               IF(hcen.LE.1.d4)THEN
                  xm2=av_leo(nb2)
               ELSE IF(hcen.GT.1.d4.AND.hcen.LE.4.d4)THEN
                  xm2=av_meo(nb2)
               ELSE
                  xm2=av_geo(nb2)
               ENDIF
c  get the area from the mass
               dam2=a_m(xm2,ia_m)
c  get the diameter from the area
               d2=dsqrt(dam2/pi)
c  total collisional cross section
               are=pi*(d1+d2)**2
c  collision probability with the two mass bins
c  For the projectiles' density I use the density densac
c  i.e. that where I subtracted the density of the
c  controlled constellation satellites (which are supposed NOT to
c  collide with the other satellites)
               prob=are*colrat(nb1,nm)*
     $  (densac(nm,nb2)*vol(nm)-kron(nb1,nb2))/(1.d0+kron(nb1,nb2))
               IF(prob.LT.0.d0)prob=0.d0
c  collision probability with particles equal or larger than nb1
c  at altitude shell  nm
```

```
               procol(nb1,nm)=procol(nb1,nm)+prob
c
               IF(prob.GT.0.d0)THEN
                  CALL xtract(prob,dt, ind)
               ELSE
                  ind=0
               ENDIF
c  generate the collision, only if it does not happen between objects
c  of the first 3 bins
               IF(ilast.EQ.1)goto 32
               IF(nb1.GT.3.or.nb2.GT.3)THEN
c  if ind=0 the loop 40 is not executed (i.e. no collision generated)
c  loop 40 on the number of collisions
                  DO 40 i=1,ind
c  discriminate between target and projectile
                     IF(xm1.GE.xm2)THEN
                        amp=xm2
                        amt=xm1
                        indp=nb2
                        indt=nb1
                     ELSE
                        amp=xm1
                        amt=xm2
                        indp=nb1
                        indt=nb2
                     ENDIF
c  EXTRACT FRAG. THRESHOLD VALUE FROM INTERVAL SS1 -- SS2
                     dn=ran2(idum)
c  Take a value between ss1 and ss2 randomly
c                 ss=ss1+(ss2-ss1)*dn
c  Take either ss1 or ss2, taking into account that,
c  considering only the population composed by payloads
c  and upper stages, 61% are payloads and 39% are rocket bodies
c  (which are harder to destroy)
                     IF(dn.LE.0.61d0)THEN
                        ss=ss1
                        sc_rb='s'
c  possibile variante (deltass va definito da qualche parte
c  (eg. nel file  collis.dat si puo' dare ss1 ss2 deltas)
c                    ss=ss1+dn*deltass
                     ELSE
                        sc_rb='r'
```

```
                      ss=ss2
c  possibile variante (deltass va definito da qualche parte
c  (eg. nel file  collis.dat si puo' dare ss1 ss2 deltas)
c                     ss=ss2-dn*deltass
                      ENDIF
c
c  orbital elements of the target (WARNING: first 2 input variables
c  are the extrema of the interval in which the inclination is choosen,
c  then impact velocity, then eccentricity, then semimajor axis)
                      vlinf=max(vcol(nm)-1.d0,0.d0)
                      CALL xtrobj(0.d0,110.d0,vcol(nm)+1.d0,vlinf,
     $                 0.d0,0.5d-1,reart+(nm*delta-delta/2.d0)+radi/2.d0,
     $                 reart+(nm*delta-delta/2.d0)-radi/2.d0,
     $                 nbin, di,vcolli,e,a, p)
                      elem(1)=a
                      elem(2)=e
                      elem(3)=di
c  random generation of the angular arguments
                      rano=ran2(idum)*360.d0
                      elem(4)=rano
                      arg=ran2(idum)*360.d0
                      elem(5)=arg
                      anom=ran2(idum)*360.d0
                      elem(6)=anom
c  generates the event
                      CALL cldgen(amt,ammin,amp,vcolli,ss,nobjx,nbin,
     $                            nsmall,'c',c2,c3,ammin0,p,ia_m,bb,bk,
     $                            n0,c6,hanada,nmom,ev_exp,se,sc_rb,
     $                            nlarg,ntot,am,ar,ncaevo,vx,vy,vz,small
     $                            ,c5)
c  eliminates the target and projectile in case of catastrophic
c  collisions, or only the projectile in case of craterizations,
c  and writes on the screen partial information
                      IF(c5.EQ.'f')THEN
c       WRITE(*,*)'************************************************'
c       WRITE(*,*)'*** one catastrophic collision happened       ***'
c       WRITE(*,*)'*** between objects in the mass bins',nb1,' and',
c    $             nb2,' ***'
c  fill the matrix with the density of the object to be removed
                         eleli(1)=elem(1)
                         eleli(2)=elem(2)
                         eleli(3)=0.d0
```

```
                        eleli(4)=0.d0
                        eleli(5)=0.d0
                        CALL dnscld(eleli,1,nhshx,indt,0, dnel)
                        DO indel=1,nhshx
                           elim(indel,indt)=elim(indel,indt)+dnel(indel)
                        ENDDO
c  the projectile is eliminated from a single altitude shell
c  since we have not its orbital elements
                           elim(nm,indp)=elim(nm,indp)+1.d0/vol(nm)
                        ELSE
c       WRITE(*,*)'************************************************'
c       WRITE(*,*)'*** one craterization happened                ***'
c       WRITE(*,*)'*** between objects in the mass bins',nb1,' and',
c    $             nb2,' ***'
c  the projectile is eliminated from a single altitude shell
c  since we have not its orbital elements
                           elim(nm,indp)=elim(nm,indp)+1.d0/vol(nm)
                        ENDIF
c  assign orbital elements to each fragment
                        IF(ntot.gt.0)THEN
                           IF(c6.eq.'q')THEN
c  if it is the output of the EVOLVE 4 model the matrix must
c  have one more dimension in order to accomodate the sampling
c  factor
c                           CALL cloud(elem,am,ar,vx,vy,vz,ncaevo,npar+1,
                           CALL cloud(elem,am,ar,vx,vy,vz,ncaevo,npar,
     $                          ntot, colbjw)
                        ELSE
c  generate artificial vector to match the calling sequence of the
c  new routine
                           DO ii=1,ntot
                              ncaevo(ii)=1
                           ENDDO
                           CALL cloud(elem,am,ar,vx,vy,vz,ncaevo,npar,
     $                       ntot, colbjw)
                        ENDIF
                     ENDIF
c  subdivide the objects in the mass bins
c  bin1  10d-3  --- 10d-2 grams
c  bin2  10d-2  --- 10d-1 grams
c  bin3  10d-1  --- 1     grams
c  bin4  1      --- 10    grams
```

```
c  bin5  10     --- 100   grams
c  bin6  100    --- 1000  grams
c  bin7  1 kg   --- 10    kg
c  bin8  10     --- 100   kg
c  bin9  100    --- 1000  kg
c  bin10 1000   --- 10000 kg
c
                    CALL zero(ncaw,nbinm)
                    IF(ntot.GT.0)THEN
                       IF(c6.eq.'q')THEN
c  objects not sampled by the EVOLVE 4 routines
                          CALL makbin(nlarg,colbjw(1:npar,1:nlarg),
     $                                xmi,xmf,nobjbx, bicw,ncaw)
c  objects sampled by the EVOLVE 4 routines
                          num_sma=ntot-nlarg
                          CALL makbevo(num_sma,
     $                                 colbjw(1:npar+1,nlarg+1:ntot),
     $                                 xmi,xmf,nobjbx, bicw,ncaw)
c  reorder matrix bin# and the vector ncamp
                          DO mi=1,nbinm
                             DO im=1,ncaw(mi)
                                DO mm=1,npar
                                  binc(mm,nca(mi)+im,mi)=bicw(mm,im,mi)
                                ENDDO
                             ENDDO
                             nca(mi)=nca(mi)+ncaw(mi)
                          ENDDO
                       ELSE
                          CALL makbin(ntot,colbjw(1:npar,1:ntot),
     $                           xmi,xmf,nobjbx, bicw,ncaw)
                       ENDIF
                    ENDIF
                    IF(c6.ne.'q')THEN
c  now assign orbital parameters to some of the small objects
                       CALL zero(nmm,nsmall)
                       DO n=1,nsmall
c  takes the integer closer to the ratio small/nsam
                          nojs=idnint(small(n)/dfloat(nsam(n)))
                          ncaw(n)=nojs
                          DO l=1,nojs
                             CALL gensma('c',xmi(n),xmf(n),elem,amp,
     $                                   vcolli,c3, sobj)
```

```
c  if the object is in orbital trajectory consider it otherwise
c  skip to the next
                              IF(sobj(1).GT.0.d0)THEN
                                 nmm(n)=nmm(n)+1
                                 DO ind=1,npar-1
                                    biw(ind,nmm(n),n)=sobj(ind)
                                 ENDDO
                              ELSE
                                 ncaw(n)=ncaw(n)-1
                              ENDIF
                           ENDDO
                        ENDDO
c  reorder the matrix bin# and the vector ncamp
                        CALL impila(biw,bicw,ncaw, binc,nca)
                     ENDIF
c  add the identification of the fragments
                     DO ibm=1,nbinm
                        DO noj=1,nca(ibm)
                           binc(npar,noj,ibm)=idcod
                        ENDDO
                     ENDDO
c  write the file  collid.ele
                     nexo=0
                     nexo1=0
                     DO lem=1,nbinm
                        nexo=nexo+ncaw(lem)*nsam(lem)
                        nexo1=nexo1+ncaw(lem)
                     ENDDO
                     IF(c5.EQ.'c')tipo='crater.'
                     IF(c5.EQ.'f')tipo='fragme.'
                     WRITE(54,33)iear,(elem(ll),ll=1,3),amt,amp,tipo,
     $                           nexo,nexo1,vcolli
c           WRITE(*,*)'***',nexo,' fragments larger than 1 mg ',
c    $                 'generated     ***'
 33                  FORMAT(i4,1x,f9.2,1x,f7.4,1x,f6.2,2(1x,f12.6),3x,
     $                   a7,1x,i9,4x,i9,7x,f7.4)
c  end of loop 40 on the number of collisions (running index i)
 40               CONTINUE
c  close the if which eliminates the collisions between small objects
               ENDIF
c  end of loop  on mass bins for the projectile (running index nb2)
 32         CONTINUE
```

```
c  end of loop  on mass bins for the target (running index nb1)
 30      CONTINUE
c
c  end of loop 31 on altitude bins (running index nm)
 31   CONTINUE
c  write the flux and the collision probability
      WRITE(55,56)iear
      WRITE(56,56)iear
 56   FORMAT('# ',i4)
      DO npro=1,nhshx
         WRITE(55,55)(rmin+delta/2.d0+(delta*(npro-1))),
     $               (colrat(ii,npro),ii=1,nbinm)
         WRITE(56,55)(rmin+delta/2.d0+(delta*(npro-1))),
     $               (procol(ii,npro),ii=1,nbinm)
 55      FORMAT(f8.1,10(1x,1pd9.3))
      ENDDO
      RETURN
      END
c=====================================================================
c
c  Version 3.0  of  28/5/2002
c
c  XTROBJ.FOR Generation of a collision/explosion
c
c   Pourpose:  generates inclination, velocity, eccentricity and
c   semimajor axis for a target object in a collision/explosion.
c   These quantities are extracted by proper probability distributions.
c
c   Input arguments:
c   AMX=maximum inclination available for the considered object;
c   AMN=minimum inclination available for the considered object;
c   VMAX=maximum velocity available for the considered object;
c   VMIN=minimum velocity available for the considered object;
c   EMAX=maximum eccentricity available for the considered object;
c   EMIN=minimum eccentricity available for the considered object;
c   AMAX=maximum smimajor axis available for the considered object;
c   AMIN=minimum smimajor axis available for the considered object;
c   P=working vector;
c   NBIN=dimension of P;
c
c   Output arguments:
c   AM=inclination (deg) of the colliding/exploding object;
```

```
c   V=velocity (Km/sec) of the colliding/exploding object;
c   E=eccentricity of the colliding/exploding object;
c   A=semimajor axis (Km) of the colliding/exploding object;
c
c   CALL subroutines: none
c
c   CALL functions: fz, ran2
c
c
c Programmer:  Alessandro Cordelli.  Dipartimento di Fisica Universita'
c di Pisa, Piazza Torricelli Nr.2, 56126 Pisa, Italy.
c E-Mail: cordelli@ipifidpt.difi.unipi.it
c
c Program modifications: none
c
c Comments: none
c
c=====================================================================
      SUBROUTINE xtrobj(amx,amn,vmax,vmin,emax,emin,amax,amin,
     $nbin, am,v,e,a, p)
      IMPLICIT none
c  INPUT
      DOUBLE PRECISION amx,amn,vmax,vmin,emax,emin,amax,amin
      INTEGER nbin
c  OUTPUT
      DOUBLE PRECISION am,v,e,a,p(nbin)
c
      DOUBLE PRECISION xmax(4),xmin(4),w(4),delta,fz,x,x1,y,ran2
      INTEGER idum,ind,i,k
c
      COMMON/rando/idum
c  LINUX VERSION
      SAVE
c
c
      xmax(1)=amx
      xmax(2)=vmax
      xmax(3)=emax
      xmax(4)=amax
c
      xmin(1)=amn
      xmin(2)=vmin
```

```
      xmin(3)=emin
      xmin(4)=amin
c
      DO ind=1,4
         delta=(xmax(ind)-xmin(ind))/dfloat(nbin)
         p(1)=0.5*(fz(xmin(ind),xmin(ind),xmax(ind),ind)
     $   +fz(xmin(ind)+delta,xmin(ind),xmax(ind),ind))*delta
         DO i=2,100
            x=xmin(ind)+(i-1)*delta
            x1=x+delta
            p(i)=0.5*(fz(x,xmin(ind),xmax(ind),ind)
     $      +fz(x1,xmin(ind),xmax(ind),ind))*delta+p(i-1)
         ENDDO
         p(nbin)=1.
         y=ran2(idum)
         IF(y.LE.p(1)) THEN
            w(ind)=delta/2.+xmin(ind)
         ELSE
            DO k=1,99
               IF(y.GT.p(k).AND.y.LE.p(k+1)) THEN
                  w(ind)=xmin(ind)+(k+0.5)*delta
                  GOTO 160
               ENDIF
            ENDDO
 160        CONTINUE
         ENDIF
      ENDDO
c
      am=w(1)
      v=w(2)
      v=w(2)
      e=w(3)
      a=w(4)
      RETURN
      END
c=======================================================================
c
c   Version   3.0  of 28/5/2002
c
c   Function FZ(W,WMIN,WMAX,IND)
c
c   Pourpose:  probability distribution for the mass, velocity,
```

```
c   eccentricity and semimajor axis of the colliding/exploding object
c
c   Input arguments:
c   W=point in which the distribution has to be calculated;
c   WMIN=minimum value for the distribution;
c   WMAX=maximum value for the distribution;
c   IND=index to identify the quantity under consideration;
c
c   Output arguments:
c   FZ=the actual value of the distribution in the point considered
c
c   CALL subroutines: none
c
c   CALL functions: none
c
c
c Programmer:  Alessandro Cordelli.  Dipartimento di Fisica Universita'
c di Pisa, Piazza Torricelli Nr.2, 56126 Pisa, Italy.
c E-Mail: cordelli@ipifidpt.difi.unipi.it
c
c Program modifications: none
c
c Comments: at the present rectangular distributions have been choosen
c
c=======================================================================
      DOUBLE PRECISION FUNCTION fz(w,wmin,wmax,ind)
      IMPLICIT none
c  INPUT
      DOUBLE PRECISION w,wmin,wmax
      INTEGER ind
c
      fz=1./(wmax-wmin)
c
      RETURN
      END
c=======================================================================
c
c  Version   3.0  of 28/5/2002
c
c  GENSMA.FOR Generation of small objects in a collision.
c
c   Pourpose:  generates a sequence of area, mass, and the three
```

```
c   cartesian components of the velocity (in the center-of-mass
c   reference frame) for sample objects in a collision
c
c   Area/Mass relation: eq. (12) WP2300
c   Distribution of fragments (collision): eqs. (38)--(40) WP2300
c   DeltaV: eq. (46)  WP2300
c
c   Input arguments:
c   T    = flag to distinguish between collision and explosions
c   XM1  = lower limit of the mass bin (Kg);
c   XM   = upper limit of the mass bin (Kg);
c   ELE  = orbital elements of the target
c   AMP  = projectile mass (Kg);
c   V    = projectile velocity (Km/sec);
c   c3   = choose between velocity models for collisions
c
c   Output arguments:
c   SOBJ=matrix with the elements, area and mass of the fragment;
c
c   CALL subroutines: cloud;
c
c   CALL functions: a_m, deltavc, ran2;
c
c
c Programmer:  Alessandro Cordelli.  Dipartimento di Fisica Universita'
c di Pisa, Piazza Torricelli Nr.2, 56126 Pisa, Italy.
c E-Mail: cordelli@ipifidpt.difi.unipi.it
c
c Program modifications: none
c
c Comments: prototype
c
c=====================================================================
c
      SUBROUTINE gensma(t,xm1,xm,ele,amp,v,c3, sobj)
      IMPLICIT none
      INCLUDE "parbin.h"
c  INPUT
      DOUBLE PRECISION xm1,xm,ele(6),amp,v
      CHARACTER c3*1,t*1
c  OUTPUT
      DOUBLE PRECISION sobj(npar+1)
```

```
c
      DOUBLE PRECISION smas(3),sarea(3),svelx(3),svely(3),svelz(3)
      DOUBLE PRECISION ran2,a_m,d,a,b,c,ep,dv,deltavc,deltav,azi,elv
      INTEGER idum,ia_m,nobb,ncaev
c
      COMMON/rando/idum
      COMMON/areama/ia_m
      INCLUDE "datbin.h"
      INCLUDE "dattri.h"
c  LINUX VERSION
      SAVE
c
c
c choose a mass value in the bin
      smas(1)=xm1+(xm-xm1)*ran2(idum)
c calculate the corresponding value for the area from the area/mass
c relation
      sarea(1)=a_m(smas(1),ia_m)
c  linear dimension of the fragment
      d=dsqrt(4.d0*sarea(1)/pi)
      IF(t.EQ.'c')THEN
c choose between low, high and intermediate velocity model
         IF(c3.EQ.'h') THEN
            a=0.2225d0
            b=-0.1022d0
            c=6.194d7
         ENDIF
         IF(c3.EQ.'l') THEN
            a=-0.9090d0
            b=-0.0868d0
            c=1.347d7
         ENDIF
         IF(c3.EQ.'i') THEN
            a=-0.125d0
            b=-0.0676d0
            c=8.01d8
         ENDIF
c kinetic energy of the projectile (in joule)
         ep=0.5d0*amp*v*v*1.d6
c   delta velocity (h model)
         dv=deltavc(d,a,b,c,v,ep)
      ELSE
```

```
         dv=deltav(d)
       ENDIF
c random directions of the velocities
       azi=twopi*ran2(idum)
       elv=dasin(2.d0*ran2(idum)-1.d0)
       svelx(1)=dv*dcos(elv)*dcos(azi)
       svely(1)=dv*dcos(elv)*dsin(azi)
       svelz(1)=dv*dsin(elv)
       nobb=1
c  generate artificial vector to match the calling sequence of the
c  new routine
       ncaev=1
       CALL cloud(ele,smas(1),sarea(1),svelx(1),svely(1),
      $svelz(1),ncaev,npar, nobb, sobj)
       IF(nobb.EQ.0)sobj(1)=0.d0
       RETURN
       END
c  ===========================================================
c    function kron
c
c  Kronecker's delta
c
c   INPUT
c    n1,n2   two integer numbers
c  ===========================================================
       INTEGER FUNCTION kron(n1,n2)
       IMPLICIT none
c  INPUT
       INTEGER n1,n2
c
       IF(n1.EQ.n2)THEN
          kron=1
       ELSE
          kron=0
       ENDIF
       RETURN
       END
c  ===========================================================
c  subroutine difmat
c
C   PURPOSE
c    Calculates the difference between two matrices
```

```
c    (if the value is lt. zero, consider zero)
C
c  input parameters
c      dm1      = first matrix
c      dm2      = second matrix
c      n1       = first dimension of the matrix
c      n2       = second dimension of the matrix
c
c  output parameters
c      dm3      =  difference matrix (dm1-dm2)
c
C   CALL SUBROUTINES
C
c   INCLUDE FILES
c
c  References
c
c  Analysis
c    A.Rossi
c  Programmer
c    A. Rossi
c
c  Modifications
c  ========================================================
      SUBROUTINE difmat(dm1,dm2,n1,n2, dm3)
      IMPLICIT none
c  INPUT
      INTEGER n1,n2
      DOUBLE PRECISION dm1(n1,n2),dm2(n1,n2)
c  OUTPUT
      DOUBLE PRECISION dm3(n1,n2)
c
      INTEGER i,j
      DO i=1,n1
         DO j=1,n2
            dm3(i,j)=max(0.d0,(dm1(i,j)-dm2(i,j)))
         ENDDO
      ENDDO
      RETURN
      END
c  ========================================================
c
```

```
c   FUNCTION   rgauss
c
C   PURPOSE
C   Generate a normally distributed random number, i.e., generate random
C   numbers with a Gaussian distribution.  These random numbers are not
C   exceptionally good -- especially in the tails of the distribution,
C   but this implementation is simple and suitable for most applications.
C   See R. W. Hamming, Numerical Methods for Scientists and Engineers,
C   McGraw-Hill, 1962, pages 34 and 389.
C
c  INPUT
c
C   XMEAN  = the mean of the Guassian distribution.
C   SD     = the standard deviation of the Guassian function
c
C   CALL SUBROUTINES: ran2
C
c   INCLUDE FILES
c
c  References
c
c  Analysis
c    A.Rossi
c  Programmer
c    FULLERTON, W., (LANL)
c
c  Modifications
c ================================================
      DOUBLE PRECISION FUNCTION RGAUSS(XMEAN,SD)
      IMPLICIT none
c  INPUT
      DOUBLE PRECISION xmean,sd,ran2
c
c  COMMON with the random number seed
      INTEGER idum,i
      COMMON/rando/idum
c
      RGAUSS = -6.0
      DO i=1,12
        RGAUSS = RGAUSS + ran2(idum)
      ENDDO
C
```

```
      RGAUSS = XMEAN + SD*RGAUSS
C
      RETURN
      END
```

A.3 CREDEN.F

```
      program creden
c  ============================================================
c   Version 3.0 of 16/4/2003
c
c   purpose
c     Read the population file and create the historical
c     densities files for SDM
c
c   CALL subroutines
c     open1 dcpmod dnsmod
c   include files
c     parcre.h altdat.h cutdat.h
c   analysis
c     A. Rossi   CPR
c   programmer
c     A. Rossi   CPR
c   program modifications
c     Introduced the possibility to read input data from a file
c     in a general FORMAT
c     (a. cordelli 24/mar/1999)
c   comments
c
c  ============================================================
      IMPLICIT none
c
      INCLUDE "parcre.h"
c  number of mass bins
      INCLUDE "parbin.h"
c  Number of parameters identifying an object
      INTEGER npa
      PARAMETER(npa=10)
c
      DOUBLE PRECISION r1(nshx),r2(nshx)
c
```

```
c  cl1--cl10: matrices with the elements of each object
c   cl1(1,n) = a
c   cl1(2,n) = e
c   cl1(3,n) = i
c   cl1(4,n) = node
c   cl1(5,n) = arg. perigee
c   cl1(6,n) = mean anomaly
c   cl1(7,n) = mass
c   cl1(8,n) = area
c   cl1(9,n) = sampling factor
c   cl1(10,n) = identification flag
c
      DOUBLE PRECISION cl1(npa,nobjx),dn1(nshx)
      DOUBLE PRECISION cl2(npa,nobjx),dn2(nshx)
      DOUBLE PRECISION cl3(npa,nobjx),dn3(nshx)
      DOUBLE PRECISION cl4(npa,nobjx),dn4(nshx)
      DOUBLE PRECISION cl5(npa,nobjx),dn5(nshx)
      DOUBLE PRECISION cl6(npa,nobjx),dn6(nshx)
      DOUBLE PRECISION cl7(npa,nobjx),dn7(nshx)
      DOUBLE PRECISION cl8(npa,nobjx),dn8(nshx)
      DOUBLE PRECISION cl9(npa,nobjx),dn9(nshx)
      DOUBLE PRECISION cl10(npa,nobjx),dn10(nshx)
c
      DOUBLE PRECISION xtemp(99),am(nbinm)
      DOUBLE PRECISION wo(npa)
      DOUBLE PRECISION fac,dm,area,dd,sma,ecc,dinc,node,ran2
      DOUBLE PRECISION sp,dmtot,dddm,ammin,ammax,delta
      DOUBLE PRECISION omperi,meanom
      CHARACTER name*9,a*4,dum*72,fmt*72,fircha*1
      INTEGER nfreq,ncl1,ncl2,ncl3,ncl4,ncl5,ncl6,ncl7,ncl8,ncl9,ncl10
      INTEGER nunit,nunit1,iniz,ntime,prop
      INTEGER nhead,ncolin,iflag,indfac,indm,inda,indd,indsma,indecc
      INTEGER indinc,indnod,indper,indmea,indac
      INTEGER nflag,reflag,idum,ifr,idat,noree
      INTEGER i,j,l,ih,ii,len
c
c  include the file with the cutoff for the masses
      INCLUDE "cutdat.h"
c  include the file with the data on the altitude shells
      INCLUDE "altdat.h"
c  include the file with the limits to switch from DCP to FOP
      INCLUDE "prop_lim.h"
```

```
c  minimum and maximum mass (used to calculate mass bins)
      data ammin,ammax/1.d-6,1.d4/
c
      sp=1
      dmtot=0.d0
c  delta between mass bins
      dddm=(ammax/ammin)**(1.d0/(dfloat(nbinm)))
c  mass bins limits in kg
      DO i=1,nbinm
c        am(im)=1.d3*ammin*(dddm**(im-1))
         am(i)=ammin*(dddm**(i-1))
      ENDDO
c
      ncl1=0
      ncl2=0
      ncl3=0
      ncl4=0
      ncl5=0
      ncl6=0
      ncl7=0
      ncl8=0
      ncl9=0
      ncl10=0
c
      delta=(rmax-rmin)/(dfloat(nhs))
      DO j=1,nhs
         r1(j)=rmin+(j-1)*delta
         r2(j)=rmin+j*delta
      ENDDO
      WRITE(*,*)'** Check the files altdat.h and cutdat.h for **'
      WRITE(*,*)'**        limits  in altitude and masses     **'
c   First year of the propagation
      READ(2,*)iniz
c   Propagation time span (in years)
      READ(2,*)ntime
c  write a file with the whole population elements every nfreq years
      READ(2,*)nfreq
c  Choice of the orbital propagator:
c        0 = only DCP
c        1 = only FOP
c        2 = DCP or FOP according to the type of orbit
      READ(2,*)prop
```

```
c  Seed of the random number generator
      READ(2,*)idum
c read the order of the columns in the input file
c     DO 123 i=1,10
c        READ(4,124) dum
c123  continue
      READ(4,*)
      READ(4,*)
 124  FORMAT(1a72)
      READ(4,124)fmt
      CALL rmbl(fmt,len)
      len=len-1
      READ(4,*)nhead
      READ(4,*)ncolin
      READ(4,*)iflag
      READ(4,*)indfac
      READ(4,*)indm
      READ(4,*)indac
      READ(4,*)indd
      READ(4,*)indsma
      READ(4,*)indecc
      READ(4,*)indinc
      READ(4,*)indnod
      READ(4,*)indper
      READ(4,*)indmea
 125  FORMAT(i2)
 126  FORMAT(8(i2))
c  Screen output for check
      WRITE(*,*)
      WRITE(*,127)nhead,ncolin
 127  FORMAT
     +   ('Skipping ',i3,' lines,  reading ',i2,' columns containing:')
      IF(iflag.NE.0)WRITE(*,*)'  Identification flag'
      IF(indfac.NE.0)WRITE(*,*)'  Sampling factor'
      IF(indm.NE.0)WRITE(*,*)'  Mass'
      IF(indac.NE.0)THEN
         IF(indac.GT.0.AND.indac.LT.100)THEN
            WRITE(*,*)'  Mass'
         ELSE IF(indac.GT.100.AND.indac.LT.200)THEN
            WRITE(*,*)'  Area over mass ratio'
            inda=indac-100
         ELSE IF(indac.GT.200.AND.indac.LT.300)THEN
```

```
            inda=indac-200
            WRITE(*,*)'  Mass over area ratio'
         ENDIF
      ENDIF
      IF(indd.NE.0)WRITE(*,*)'  Diameter'
      IF(indsma.NE.0)WRITE(*,*)'  Semimajor axis'
      IF(indecc.NE.0)WRITE(*,*)'  Eccentricity'
      IF(indinc.NE.0)WRITE(*,*)'  Inclination'
      IF(indnod.NE.0)WRITE(*,*)'  Righ ascension of the node'
      IF(indper.NE.0)WRITE(*,*)'  Argument of perigee'
      IF(indmea.NE.0)WRITE(*,*)'  Mean anomaly'
      WRITE(*,*)
c  read the population file
c
c  if iflag and/or indfac are set to zero a default value for the
c  ID of the object/sampling factor is assumed
c     IF(nhead.ne.0) THEN
c  Read the epoch of the elements ,i.e. the first year
c  of the propagation
      READ(1,*)iniz
c  Skip header lines
      DO i=1,nhead
         READ(1,*)
      ENDDO
c     ENDIF
c.................
      nflag=0
c.................
      DO i=1,ntotx
c  check for comment lines
         READ(1,'(a)',end=11)fircha
         IF(fircha.ne.'#')THEN
            backspace(1)
c  read identification flag, sampling factor,mass (kg) ,area (m**2),
c  diameter (m), semimajor axis (km) ,eccentricity,
c  inclination (degrees)
            IF(fmt(1:len).eq.'*') THEN
               READ(1,*,end=11)(xtemp(j),j=1,ncolin)
            ELSE
               READ(1,fmt,end=11)(xtemp(j),j=1,ncolin)
            ENDIF
c.....................................
```

```
            IF(iflag.eq.0) THEN
               nflag=nflag+1
            ELSE
               nflag=xtemp(iflag)
            ENDIF
            IF(indfac.eq.0) THEN
               fac=1
            ELSE
               fac=xtemp(indfac)
            ENDIF
c.....................................
c  mass
            dm=xtemp(indm)
c  area: must distiguish between:
c        0 < indac < 100         files including area itself
c        100 < indac < 200       files including area over mass
c        200 < indac < 300       files including mass over area
            IF(indac.GT.0.and.indac.LT.100)THEN
               area=xtemp(inda)
            ELSE IF(indac.GT.100.and.indac.LT.200)THEN
               area=xtemp(inda)*dm
            ELSE IF(indac.GT.200.and.indac.LT.300)THEN
               area=dm/xtemp(inda)
            ENDIF
c  diameter (not a compulsory parameter)
            IF(indd.NE.0)dd=xtemp(indd)
c  semi major axis
            sma=xtemp(indsma)
c  eccentricity
            ecc=xtemp(indecc)
c  inclination
            dinc=xtemp(indinc)
c  patch to avoid problems in FOP
            if(dinc.eq.0.d0)dinc=1.d-4
c  Random generation of angular elements if not existent in
c  the population file
c  node
            IF(indnod.NE.0)THEN
               node=xtemp(indnod)
            ELSE
               WRITE(*,*)
     +              'WARNING: NODE not read from the population file'
```

```
               WRITE(*,*)'**** Generating a random value ****'
               node=ran2(idum)*360.d0
            ENDIF
c  argument of perigee
            IF(indper.NE.0)THEN
               omperi=xtemp(indper)
            ELSE
               WRITE(*,*)
     + 'WARNING: ARGUMENT OF PERIGEE not read from the population file'
               WRITE(*,*)'**** Generating a random value ****'
               omperi=ran2(idum)*360.d0
            ENDIF
c  mean anomaly
            IF(indmea.NE.0)THEN
               meanom= xtemp(indmea)
            ELSE
               WRITE(*,*)
     +        'WARNING: MEAN ANOMALY not read from the population file'
               WRITE(*,*)'**** Generating a random value ****'
               meanom=ran2(idum)*360.d0
            ENDIF
c  area over mass in km**2/kg
c            daom=(area/dm)/1.d+6
c  mass in g
c           dm=dm*1000.d0
c  consider only the masses larger than am(1) and lower than the
c  cutoff mass contained in the file cutdat.h
            IF(dm.le.dlarg.and.dm.ge.am(1))THEN
               IF(dm.ge.am(1).and.dm.lt.am(2))THEN
                  ncl1=ncl1+1
                  cl1(1,ncl1)=sma
                  cl1(2,ncl1)=ecc
                  cl1(3,ncl1)=dinc
                  cl1(4,ncl1)=node
                  cl1(5,ncl1)=omperi
                  cl1(6,ncl1)=meanom
                  cl1(7,ncl1)=area
                  cl1(8,ncl1)=dm
                  cl1(9,ncl1)=fac
                  cl1(10,ncl1)=nflag
               ENDIF
               IF(dm.ge.am(2).and.dm.lt.am(3))THEN
```

```
      ncl2=ncl2+1
      cl2(1,ncl2)=sma
      cl2(2,ncl2)=ecc
      cl2(3,ncl2)=dinc
      cl2(4,ncl2)=node
      cl2(5,ncl2)=omperi
      cl2(6,ncl2)=meanom
      cl2(7,ncl2)=area
      cl2(8,ncl2)=dm
      cl2(9,ncl2)=fac
      cl2(10,ncl2)=nflag
   ENDIF
    IF(dm.ge.am(3).and.dm.lt.am(4))THEN
      ncl3=ncl3+1
      cl3(1,ncl3)=sma
      cl3(2,ncl3)=ecc
      cl3(3,ncl3)=dinc
      cl3(4,ncl3)=node
      cl3(5,ncl3)=omperi
      cl3(6,ncl3)=meanom
      cl3(7,ncl3)=area
      cl3(8,ncl3)=dm
      cl3(9,ncl3)=fac
      cl3(10,ncl3)=nflag
    ENDIF
    IF(dm.ge.am(4).and.dm.lt.am(5))THEN
      ncl4=ncl4+1
      cl4(1,ncl4)=sma
      cl4(2,ncl4)=ecc
      cl4(3,ncl4)=dinc
      cl4(4,ncl4)=node
      cl4(5,ncl4)=omperi
      cl4(6,ncl4)=meanom
      cl4(7,ncl4)=area
      cl4(8,ncl4)=dm
      cl4(9,ncl4)=fac
      cl4(10,ncl4)=nflag
    ENDIF
    IF(dm.ge.am(5).and.dm.lt.am(6))THEN
      ncl5=ncl5+1
      cl5(1,ncl5)=sma
      cl5(2,ncl5)=ecc
```

```
      cl5(3,ncl5)=dinc
      cl5(4,ncl5)=node
      cl5(5,ncl5)=omperi
      cl5(6,ncl5)=meanom
      cl5(7,ncl5)=area
      cl5(8,ncl5)=dm
      cl5(9,ncl5)=fac
      cl5(10,ncl5)=nflag
   ENDIF
   IF(dm.ge.am(6).and.dm.lt.am(7))THEN
      ncl6=ncl6+1
      cl6(1,ncl6)=sma
      cl6(2,ncl6)=ecc
      cl6(3,ncl6)=dinc
      cl6(4,ncl6)=node
      cl6(5,ncl6)=omperi
      cl6(6,ncl6)=meanom
      cl6(7,ncl6)=area
      cl6(8,ncl6)=dm
      cl6(9,ncl6)=fac
      cl6(10,ncl6)=nflag
   ENDIF
   IF(dm.ge.am(7).and.dm.lt.am(8))THEN
      ncl7=ncl7+1
      cl7(1,ncl7)=sma
      cl7(2,ncl7)=ecc
      cl7(3,ncl7)=dinc
      cl7(4,ncl7)=node
      cl7(5,ncl7)=omperi
      cl7(6,ncl7)=meanom
      cl7(7,ncl7)=area
      cl7(8,ncl7)=dm
      cl7(9,ncl7)=fac
      cl7(10,ncl7)=nflag
  ENDIF
   IF(dm.ge.am(8).and.dm.lt.am(9))THEN
      ncl8=ncl8+1
      cl8(1,ncl8)=sma
      cl8(2,ncl8)=ecc
      cl8(3,ncl8)=dinc
      cl8(4,ncl8)=node
      cl8(5,ncl8)=omperi
```

```
                  cl8(6,ncl8)=meanom
                  cl8(7,ncl8)=area
                  cl8(8,ncl8)=dm
                  cl8(9,ncl8)=fac
                  cl8(10,ncl8)=nflag
               ENDIF
               IF(dm.ge.am(9).and.dm.lt.am(10))THEN
                  ncl9=ncl9+1
                  cl9(1,ncl9)=sma
                  cl9(2,ncl9)=ecc
                  cl9(3,ncl9)=dinc
                  cl9(4,ncl9)=node
                  cl9(5,ncl9)=omperi
                  cl9(6,ncl9)=meanom
                  cl9(7,ncl9)=area
                  cl9(8,ncl9)=dm
                  cl9(9,ncl9)=fac
                  cl9(10,ncl9)=nflag
               ENDIF
               IF(dm.ge.am(10))THEN
                  ncl10=ncl10+1
                  cl10(1,ncl10)=sma
                  cl10(2,ncl10)=ecc
                  cl10(3,ncl10)=dinc
                  cl10(4,ncl10)=node
                  cl10(5,ncl10)=omperi
                  cl10(6,ncl10)=meanom
                  cl10(7,ncl10)=area
                  cl10(8,ncl10)=dm
                  cl10(9,ncl10)=fac
                  cl10(10,ncl10)=nflag
               ENDIF
            ENDIF
         ENDIF
      ENDDO
 11   CONTINUE
ccccc
      WRITE(*,*)
      WRITE(*,*)'NUMBER OF OBJECTS IN THE MASS BINS '
      WRITE(*,*)ncl1,ncl2,ncl3,ncl4,ncl5,ncl6,ncl7,ncl8,ncl9,ncl10
      WRITE(*,*)
ccccc
```

```
c  loop on the propagation time span
      ifr=0
      DO l=1,ntime
c  CALL the DCP
         idat=iniz+l-1
         WRITE(*,*)idat
         nunit=11
         WRITE(a,13)idat
 13      FORMAT(i4)
         name='dens.dat'
         CALL open1(nunit,name,a,'unknown')
c  Propagation
c  Mass Bin 1
         IF(l.ne.1)THEN
            noree=0
            DO i=1,ncl1
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl1(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl1(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl1(1,i).gt.lima.or.cl1(2,i).gt.lime)THEN
                       CALL fop(idat,sp,1,cl1(1,i) ,reflag,wo)


                     WRITE(*,*)'FOP'

                  ELSE
                     CALL dcpmod(idat,sp,1,cl1(1,i), reflag,wo)

                     WRITE(*,*)'DCP'

                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
                     cl1(j,noree)=wo(j)
                  ENDDO
               ENDIF
            ENDDO
```

```
            ncl1=noree
          ENDIF
          CALL dnsmod(cl1,ncl1,nhs,npa, dn1)
c  Mass Bin 2
          IF(l.ne.1)THEN
            noree=0
            DO i=1,ncl2
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl2(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl2(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl2(1,i).gt.lima.or.cl2(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl2(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl2(1,i) ,reflag,wo)
                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
                     cl2(j,noree)=wo(j)
                  ENDDO
               ENDIF
            ENDDO
            ncl2=noree
          ENDIF
          CALL dnsmod(cl2,ncl2,nhs,npa, dn2)
c  Mass Bin 3
          IF(l.ne.1)THEN
            noree=0
            DO i=1,ncl3
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl3(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl3(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl3(1,i).gt.lima.or.cl3(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl3(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl3(1,i) ,reflag,wo)
```

```
                ENDIF
              ENDIF
c  If the object did not reenter load the matrix
              IF(reflag.EQ.1)THEN
                noree=noree+1
                DO j=1,npa
                   cl3(j,noree)=wo(j)
                ENDDO
              ENDIF
           ENDDO
           ncl3=noree
         ENDIF
         CALL dnsmod(cl3,ncl3,nhs,npa, dn3)
c  Mass Bin 4
         IF(l.ne.1)THEN
           noree=0
           DO i=1,ncl4
              IF(prop.eq.0)THEN
                 CALL dcpmod(idat,sp,1,cl4(1,i), reflag,wo)
              ELSE IF(prop.eq.1)THEN
                 CALL fop(idat,sp,1,cl4(1,i) ,reflag,wo)
              ELSE IF(prop.eq.2)THEN
                 IF(cl4(1,i).gt.lima.or.cl4(2,i).gt.lime)THEN
                    CALL dcpmod(idat,sp,1,cl4(1,i), reflag,wo)
                 ELSE
                    CALL fop(idat,sp,1,cl4(1,i) ,reflag,wo)
                 ENDIF
              ENDIF
c  If the object did not reenter load the matrix
              IF(reflag.EQ.1)THEN
                 noree=noree+1
                 DO j=1,npa
                    cl4(j,noree)=wo(j)
                 ENDDO
              ENDIF
           ncl4=noree
           ENDDO
         ENDIF
         CALL dnsmod(cl4,ncl4,nhs,npa, dn4)
c  Mass Bin 5
         IF(l.ne.1)THEN
           noree=0
```

```
            DO i=1,ncl5
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl5(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl5(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl5(1,i).gt.lima.or.cl5(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl5(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl5(1,i) ,reflag,wo)
                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
                     cl5(j,noree)=wo(j)
                  ENDDO
               ENDIF
            ENDDO
            ncl5=noree
         ENDIF
         CALL dnsmod(cl5,ncl5,nhs,npa, dn5)
c  Mass Bin 6
         IF(l.ne.1)THEN
            noree=0
            DO i=1,ncl6
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl6(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl6(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl6(1,i).gt.lima.or.cl6(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl6(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl6(1,i) ,reflag,wo)
                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
```

```
                  cl6(j,noree)=wo(j)
               ENDDO
            ENDIF
         ENDDO
         ncl6=noree
      ENDIF
      CALL dnsmod(cl6,ncl6,nhs,npa, dn6)
c  Mass Bin 7
      IF(l.ne.1)THEN
         noree=0
         DO i=1,ncl7
            IF(prop.eq.0)THEN
               CALL dcpmod(idat,sp,1,cl7(1,i), reflag,wo)
            ELSE IF(prop.eq.1)THEN
               CALL fop(idat,sp,1,cl7(1,i) ,reflag,wo)
            ELSE IF(prop.eq.2)THEN
               IF(cl7(1,i).gt.lima.or.cl7(2,i).gt.lime)THEN
                  CALL dcpmod(idat,sp,1,cl7(1,i), reflag,wo)
               ELSE
                  CALL fop(idat,sp,1,cl7(1,i) ,reflag,wo)
               ENDIF
            ENDIF
c  If the object did not reenter load the matrix
            IF(reflag.EQ.1)THEN
               noree=noree+1
               DO j=1,npa
                  cl7(j,noree)=wo(j)
               ENDDO
            ENDIF
         ENDDO
         ncl7=noree
      ENDIF
      CALL dnsmod(cl7,ncl7,nhs,npa, dn7)
c  Mass Bin 8
      IF(l.ne.1)THEN
         noree=0
         DO i=1,ncl8
            IF(prop.eq.0)THEN
               CALL dcpmod(idat,sp,1,cl8(1,i), reflag,wo)
            ELSE IF(prop.eq.1)THEN
               CALL fop(idat,sp,1,cl8(1,i) ,reflag,wo)
            ELSE IF(prop.eq.2)THEN
```

```
                  IF(cl8(1,i).gt.lima.or.cl8(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl8(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl8(1,i) ,reflag,wo)
                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
                     cl8(j,noree)=wo(j)
                  ENDDO
               ENDIF
            ENDDO
            ncl8=noree
         ENDIF
         CALL dnsmod(cl8,ncl8,nhs,npa, dn8)
c  Mass Bin 9
         IF(l.ne.1)THEN
            noree=0
            DO i=1,ncl9
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl9(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl9(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl9(1,i).gt.lima.or.cl9(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl9(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl9(1,i) ,reflag,wo)
                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
                     cl9(j,noree)=wo(j)
                  ENDDO
               ENDIF
            ENDDO
            ncl9=noree
         ENDIF
```

```
         CALL dnsmod(cl9,ncl9,nhs,npa, dn9)
c  Mass Bin 10
         IF(l.ne.1)THEN
            noree=0
            DO i=1,ncl10
               IF(prop.eq.0)THEN
                  CALL dcpmod(idat,sp,1,cl10(1,i), reflag,wo)
               ELSE IF(prop.eq.1)THEN
                  CALL fop(idat,sp,1,cl10(1,i) ,reflag,wo)
               ELSE IF(prop.eq.2)THEN
                  IF(cl10(1,i).gt.lima.or.cl10(2,i).gt.lime)THEN
                     CALL dcpmod(idat,sp,1,cl10(1,i), reflag,wo)
                  ELSE
                     CALL fop(idat,sp,1,cl10(1,i) ,reflag,wo)
                  ENDIF
               ENDIF
c  If the object did not reenter load the matrix
               IF(reflag.EQ.1)THEN
                  noree=noree+1
                  DO j=1,npa
                     cl10(j,noree)=wo(j)
                  ENDDO
               ENDIF
            ENDDO
            ncl10=noree
         ENDIF
         CALL dnsmod(cl10,ncl10,nhs,npa, dn10)
c  write header
         WRITE(nunit,3)
 3    FORMAT('#Initial altitude  Final altitude  Shells width [km]')
         WRITE(nunit,4)
 4    FORMAT(  '#0                  40000               50'   )
         DO ih=1,nhs
            WRITE(nunit,17)dn1(ih),dn2(ih),dn3(ih),dn4(ih),
     +       dn5(ih),dn6(ih),dn7(ih),dn8(ih),dn9(ih),dn10(ih)
         ENDDO
 17      FORMAT(1pd9.3,9(1x,1pd9.3))
         close(nunit)
         IF(ifr.eq.nfreq)THEN
            name='popu.dat'
            nunit1=31
            CALL open1(nunit1,name,a,'unknown')
```

```
c  write header
           WRITE(nunit1,100)
 100       FORMAT('# id     sampling          a       e       i     ',
     +            '   Node     Arg.Per.   M     Area [m^2]  Mass [kg]')
           DO ii=1,ncl1
c  id, sampling, a, e, i, area [m^2], mass [kg]
              WRITE(nunit1,31)idint(cl1(10,ii)),cl1(9,ii),
     +              cl1(1,ii),cl1(2,ii),cl1(3,ii),cl1(4,ii),cl1(5,ii),
     +              cl1(6,ii),cl1(7,ii),cl1(8,ii)
           ENDDO
           DO ii=1,ncl2
              WRITE(nunit1,31)idint(cl2(10,ii)),cl2(9,ii),
     +              cl2(1,ii),cl2(2,ii),cl2(3,ii),cl2(4,ii),cl2(5,ii),
     +              cl2(6,ii),cl2(7,ii),cl2(8,ii)
           ENDDO
           DO ii=1,ncl3
              WRITE(nunit1,31)idint(cl3(10,ii)),cl3(9,ii),
     +              cl3(1,ii),cl3(2,ii),cl3(3,ii),cl3(4,ii),cl3(5,ii),
     +              cl3(6,ii),cl3(7,ii),cl3(8,ii)
           ENDDO
           DO ii=1,ncl4
              WRITE(nunit1,31)idint(cl4(10,ii)),cl4(9,ii),
     +              cl4(1,ii),cl4(2,ii),cl4(3,ii),cl4(4,ii),cl4(5,ii),
     +              cl4(6,ii),cl4(7,ii),cl4(8,ii)
           ENDDO
           DO ii=1,ncl5
              WRITE(nunit1,31)idint(cl5(10,ii)),cl5(9,ii),
     +              cl5(1,ii),cl5(2,ii),cl5(3,ii),cl5(4,ii),cl5(5,ii),
     +              cl5(6,ii),cl5(7,ii),cl5(8,ii)
           ENDDO
           DO ii=1,ncl6
              WRITE(nunit1,31)idint(cl6(10,ii)),cl6(9,ii),
     +              cl6(1,ii),cl6(2,ii),cl6(3,ii),cl6(4,ii),cl6(5,ii),
     +              cl6(6,ii),cl6(7,ii),cl6(8,ii)
           ENDDO
           DO ii=1,ncl7
              WRITE(nunit1,31)idint(cl7(10,ii)),cl7(9,ii),
     +              cl7(1,ii),cl7(2,ii),cl7(3,ii),cl7(4,ii),cl7(5,ii),
     +              cl7(6,ii),cl7(7,ii),cl7(8,ii)
           ENDDO
           DO ii=1,ncl8
              WRITE(nunit1,31)idint(cl8(10,ii)),cl8(9,ii),
```

```
     +               cl8(1,ii),cl8(2,ii),cl8(3,ii),cl8(4,ii),cl8(5,ii),
     +               cl8(6,ii),cl8(7,ii),cl8(8,ii)
            ENDDO
            DO ii=1,ncl9
               WRITE(nunit1,31)idint(cl9(10,ii)),cl9(9,ii),
     +               cl9(1,ii),cl9(2,ii),cl9(3,ii),cl9(4,ii),cl9(5,ii),
     +               cl9(6,ii),cl9(7,ii),cl8(8,ii)
            ENDDO
            DO ii=1,ncl10
               WRITE(nunit1,31)idint(cl10(10,ii)),cl10(9,ii),
     +           cl10(1,ii),cl10(2,ii),cl10(3,ii),cl10(4,ii),cl10(5,ii),
     +           cl10(6,ii),cl10(7,ii),cl10(8,ii)
            ENDDO
            ifr=0
            close(nunit1)
         ENDIF
c  counter for population writing
         ifr=ifr+1
 31      FORMAT(i9,1x,e10.4,1x,f11.2,1x,f6.4,1x,f7.2,3(1x,f9.2),
     +          2(1x,e10.4))
       ENDDO
       END
```

A.4 DCP.F

```
      SUBROUTINE dcp(iear,sp,ntot,w, nflag,a,e,di,ar,am)
c=====================================================================
C   Version of 13/6/03
C   PURPOSE
C    The Debris Cloud Propagator propagates a cloud of objects
C    given the semimajor axis, eccentricity, inclination and
C    ballistic parameter. The perturbation acting over the
C    objects is only the atmospheric drag.
C
c  input parameters
c      iear     =  year of the propagation
c      sp       =  time span of the propagation in years
c                  (es. 1.3d0 = 365 + 365*0.3 days)
c      ntot     =  total number of objects to be propagated
c      w        =  matrix with orbital elements and
c                  area (m**2) and mass (kg) of all the objects
c  output parameters
```

```
c      nflag    flag to check if the object reentered:
c                  nflag = 1   no reenter
c                  nflag = 0   reenter
c      a        semimajor axis after propagation (km)
c      e        eccentricity after propagation
c      di       inclination after propagation (degrees)
c      ar       area of the object (m**2)
c      am       mass of the objects (kg)
C
C   CALL SUBROUTINES
C    finste, solflux, period, td88m, d_a, d_e, restim
C
c   INCLUDE FILES
c    pardcp.h, dattri.h, datear.h, datdcp.h
c
c  References
c    Study Note of WP 3200
c
c  Analysis
c    L. Anselmo, A.Rossi
c  Programmer
c    A.Rossi - CPR - PISA
c
c  Modifications
c    Air densty multiplied by 2 in order to better match reentry
c    tests (check!!!!!!!!!!!!!!!)
c
c  Comments
c    Only the apogee is supposed to decrease until the orbit
c    becomes quasi-circular.
c    If an object decays we put semimajor axis and mass equal
c    to zero, so the object does not contribute to the density
c
c=======================================================================
      IMPLICIT none
      INCLUDE "pardcp.h"
      INCLUDE 'parbin.h'
c  INPUT
      INTEGER iear,ntot
      DOUBLE PRECISION sp,w(npar-1)
c  OUTPUT
      INTEGER nflag
```

```
      DOUBLE PRECISION a(nfragx),e(nfragx),di(nfragx)
      DOUBLE PRECISION ar(nfragx),am(nfragx)
c
      DOUBLE PRECISION rp(nfragx),ro(nfragx),scale(nfragx),ra(nfragx)
      DOUBLE PRECISION deltat(nfragx),aom(nfragx),del(nfragx)
      DOUBLE PRECISION vperi(nfragx),per(nfragx),deltaa(nfragx)
      DOUBLE PRECISION deltae(nfragx),bi(nfragx),dsp,step,span
      DOUBLE PRECISION om_ea,dlom,flux
      INTEGER iw,niter,ncount,n,lt,i
c
      INCLUDE "dattri.h"
      INCLUDE "datear.h"
      INCLUDE "datdcp.h"
c  LINUX VERSION
      save
c
      nflag=0
ccc
c
c           write(97,97)iear,(w(ii),ii=1,5),nflag
c97         format('INIZO ',i5,5(1x,f20.8),i5)
c
ccc
c  create working vectors
      DO 1 iw=1,ntot
         a(iw)=w(1)
         e(iw)=w(2)
         di(iw)=w(3)*pi/180.d0
         ar(iw)=w(npar_orb+1)
         am(iw)=w(npar_orb+2)
 1    CONTINUE
c  find the step for the propagation
      dsp=sp*365.d0
      CALL finste(dsp,step,niter)
c  time span of action of the drag
      span=step*86400.d0
      ncount=0
c  angular velocity of the Earth in rad/sec
      om_ea=twopi/86164.d0
      dlom=dlambd*om_ea
c
      DO n=1,ntot
```

```
c  Convert the ballistic coefficient to km**2/kg
c  (WARNING: we also have to multiply by two due to the definition
c  of the ballistic coefficient coming from the TLOE)
         rp(n)=a(n)*(1.d0-e(n))
         ra(n)=a(n)*(1.d0+e(n))
c  the area/mass has to be in km**2/kg
         aom(n)=(ar(n)/am(n))*1.d-6
c  ballistic coefficient
         bi(n)=aom(n)*cd
      ENDDO
c  loop on time (running index   lt  )
      DO lt=1,niter
c  calculation of the solax flux at 10.7 cm
c        aic=aic+step*(lt-1)/365.25d0
         aic=(iear-dminfl)+step*(lt-1)/365.25d0
         CALL solflux(aic, flux)
c  loop  on number of objects (running index  i  )
         DO i=1,ntot
            ncount=ncount+1
            ra(i)=2.d0*a(i)-rp(i)
c  if the perigee is over the upper limit of the atmosphere
c  the drag DOes not act, so the object orbit does not change
            IF(rp(i).gt.rmxdcp)GOTO 40
c  if the perigee is under the lowest permitted altitude it
c  is considered reentered and the object i is skipped
            IF(rp(i).lt.rmi.or.ra(i).lt.rmi)THEN
               ncount=ncount-1
               a(i)=0.d0
               am(i)=0.d0
               GOTO 40
            ENDIF
c  period of revolution
            CALL period(a(i), per(i))
            IF((ra(i)-rp(i)).le.epsi)THEN
c  circular orbit, rp start to decrease
c
c  if the orbit is near circular, i.e. r_apogee-p_perigee.lt.2*scale,
c  then the residence time is the whole period of revolution and
c  we DOn't need to calculate it
               deltat(i)=span
c  in this case the density is the one at the heigth of the semimajor axis
               CALL td88m(flux,(a(i)-reart), ro(i),scale(i))
```

```
c  Adjust the density according to observed values
                  ro(i)=ro(i)*2.d0
c  in this case the velocity is the circular velocity
               vperi(i)=dsqrt(ge/a(i))
c
              del(i)=bi(i)*(1.d0-(dlom*dcos(di(i))*rp(i)/vperi(i)))**2.d0
               CALL d_a(a(i),e(i),vperi(i),ro(i),deltat(i),
     +                   del(i), deltaa(i))
               CALL d_e(e(i),vperi(i),ro(i),del(i),deltat(i),span,
     +          per(i), deltae(i))
               a(i)=a(i)+deltaa(i)
               e(i)=e(i)+deltae(i)
c  avoid negative eccentricities
               IF(e(i).lt.0.d0)e(i)=0.d0
               rp(i)=a(i)*(1.d0-e(i))
               IF(lt.eq.niter.and.(a(i)*(1-e(i))).lt.rmi)THEN
                  a(i)=0.d0
                  am(i)=0.d0
               ENDIF
            ELSE
c  atmospheric data (density and scale height)
               CALL td88m(flux,(rp(i)-reart), ro(i),scale(i))
c  Adjust the density according to observed values
                  ro(i)=ro(i)*2.d0
c  elliptic orbit, perigee is fixed and only the semimajor axis
c  is supposed to decrease
               IF((ra(i)-rp(i)).le.2.d0*scale(i))THEN
c  low eccentricity
                  deltat(i)=span
c  in this case the density is the one at the heigth of the semimajor axis
                  CALL td88m(flux,(a(i)-reart), ro(i),scale(i))
c  Adjust the density
                  ro(i)=ro(i)*2.d0
c
c  in this case the velocity is the circular velocity
                  vperi(i)=dsqrt(ge/a(i))
c  the density is a mean between the density at rp and ra
                  ro(i)=(scale(i)/(ra(i)-rp(i)))*ro(i)*
     +            (1.d0-exp(-(ra(i)-rp(i))/scale(i)))
c
                  del(i)=bi(i)*(1.d0-(dlom*dcos(di(i))
     +                  *rp(i)/vperi(i)))**2.d0
```

```
                  CALL d_a(a(i),e(i),vperi(i),ro(i),deltat(i),
     +                    del(i), deltaa(i))
                  a(i)=a(i)+deltaa(i)
                  e(i)=1.d0-(rp(i)/a(i))
c  avoid negative eccentricities
                  IF(e(i).lt.0.d0)e(i)=0.d0
                  IF(lt.eq.niter.and.(a(i)*(1.d0-e(i))).lt.rmi)THEN
                     a(i)=0.d0
                     am(i)=0.d0
                  ENDIF
               ELSE
c  high eccentricity
c  the density is a mean between the density at rp and at rp+2*scale
                  ro(i)=ro(i)*0.432332d0
c  perigee velocity
                  vperi(i)=dsqrt((ge/a(i))*((1+e(i))/(1-e(i))))
c
             del(i)=bi(i)*(1.d0-(dlom*dcos(di(i))*rp(i)/vperi(i)))**2.d0
c  calculation of the fraction of the orbit spent in the atmosphere
                  CALL restim(e(i),2.d0*scale(i)/a(i), deltat(i))
                  deltat(i)=deltat(i)*span
                  CALL d_a(a(i),e(i),vperi(i),ro(i),deltat(i),
     +                    del(i), deltaa(i))
                  a(i)=a(i)+deltaa(i)
                  e(i)=1.d0-(rp(i)/a(i))
c  avoid negative eccentricities
                  IF(e(i).lt.0.d0)e(i)=0.d0
                  IF(lt.eq.niter.and.(a(i)*(1.d0-e(i))).lt.rmi)THEN
                     a(i)=0.d0
                     am(i)=0.d0
                  ENDIF
               ENDIF
            ENDIF
c
 40         CONTINUE
c  end of loop on number of objects (running index  i  )
         ENDDO
c  end of loop on time (running index   lt  )
      ENDDO
      IF(a(1).ge.rmi)nflag=1
c  change back to degrees
      DO i=1,ntot
```

```
         di(i)=di(i)*180.d0/pi
      ENDDO
ccc
c
c       IF(nflag.eq.0)write(97,98)iear,w(1),a,e,di,ar,am,nflag
c           write(97,98)iear,a(1),e(1),di(1),ar(1),am(1),nflag
c98      format('FINE ',i5,5(1x,f20.8),i5)
c
ccc
      RETURN
      END
c=======================================================================
```

A.5 LAUSUB.F

```
c  ====================================================================
c   SUBROUTINE CONSTE
c   Version 3.0    of  3/7/2003
c
c  PURPOSE
c  Simulates the building of a constellation
c
c  INPUT ARGUMENTS
c     iear      year of the simulation
c     deflag    flag for de-orbiting options
c                 c  = circular orbit
c                 e  = elliptical orbit
c     reslif    residual lifetime of the orbit in which the deorbited
c                satellites are placed
c     perdeo    perigee altitude of the spacecraft which are
c               re-orbited in a LEO graveyard orbit
c     deoalt    altitude of the LEO circular graveyard orbit
c      h1,h2    Amplitudes of the triangular distribution for
c               deorbiting
c
c  OUTPUT ARGUMENTS
c      nconst    number of constellations to be launched
c      nla       total number of objects injected in orbit
c                in the year  iear   due to constellations
c      nlanci    number of launches in the iear  year
c      cobj      matrix with the objects launched in the constellations
c
```

```
c  CALL SUBROUTINES
c    reacon
c
c  INCLUDE FILES
c    parlau.h, paralt.h, parbin.h, stacon.h, idcode.h
c
c  REFERENCE
c     Study Notes of the WP2200
c
c  ANALYSIS
c   A. Rossi
c  PROGRAMMER
c   A. Rossi
c  PROGRAM MODIFICATIONS
c   none
c  COMMENTS
c   UNIX version
c
c  ================================================================
      SUBROUTINE conste(iear,deflag,reslif,perdeo,deoalt,au_fi,h1,h2,
     +                  nconst,nla,nlanci,cobj)
c
      IMPLICIT none
      INCLUDE "parlau.h"
      INCLUDE "paralt.h"
      INCLUDE "parbin.h"
c  INPUT
      INTEGER iear,nconst,nla,nlanci
c  matrix with the objects launched in the constellations
c  (semimajor axis, eccentricity, inclination, mass)
      DOUBLE PRECISION reslif,perdeo,deoalt,cobj(npar,nmax)
      CHARACTER deflag*1,au_fi*1
      DOUBLE PRECISION h1,h2
c
      LOGICAL first
c
      DOUBLE PRECISION areaco(nconstx),dmasco(nconstx)
      DOUBLE PRECISION eleco(npar_orb,nconstx),eleusco(npar_orb,nconstx)
      DOUBLE PRECISION areuco(nconstx),dmauco(nconstx),fail(nconstx)
      DOUBLE PRECISION obj(nconstx),wv(npar),wv0(npar),a_m
      DOUBLE PRECISION deo_ele(npar_orb,nconstx)
      INTEGER iearco(nconstx),life(nconstx),nsatt(nconstx)
```

```
      INTEGER neach(nconstx),la1(nconstx),la2(nconstx)
      INTEGER ndeo(nconstx),ieaupc(nconstx),nsat(nconstx),necy(nconstx)
      INTEGER nobr(nconstx),nupc(nconstx),deo_pol(nconstx)
      INTEGER ia_m,i,icon,lanci,ifa,l,ii,nfai,j
c      dimension deli(nhshx),dnn(nhshx)
c
c      COMMON/elimin/elim(nhshx,nbinm)
      COMMON/areama/ia_m
c
      INCLUDE "stacon.h"
      INCLUDE "idcode.h"
      DATA first/.true./
c  LINUX VERSION
      save
c
      nlanci=0
      nla=0
      nfai=0
c
      IF(first)THEN
c  file with the characteristic of the constellations (unit 4)
c  read the file launch.in
c  # of constellations, # of sat/const, a, e, i, mass of a satellite
c  launches/year to built, launches/year to maintain
         CALL reacon(4, nconst,iearco,life,nsatt,neach,la1,la2,
     +               areaco,dmasco,eleco,ndeo,deo_pol,deo_ele,
     +               fail,ieaupc,areuco,dmauco,eleusco,obj)
         first=.false.
      ENDIF
c  loop on all the simulated constellations
      DO 10 i=1,nconst
c  working vector with the constellation elements and characteristics
         DO j=1,npar_orb
            wv(j)=eleco(j,i)
         ENDDO
         wv(npar_orb+1)=areaco(i)
         wv(npar_orb+2)=dmasco(i)
         wv(npar)=idcoco
c  flag defining the type of satellite (controlled or not controlled)
         icon=0
c  number of years necessary to build the constellation
         IF(mod(nsatt(i),(la1(i)*neach(i))).eq.0)THEN
```

```
          necy(i)=idint(dfloat(nsatt(i))/dfloat(la1(i)*neach(i)))
        ELSE
          necy(i)=idint(dfloat(nsatt(i))/dfloat(la1(i)*neach(i)))+1
        ENDIF
c  if the year iear is later than the END of lifetime of the
c  constellation there are three cases
        IF(iear.gt.(iearco(i)+life(i)))THEN
           IF(life(i).lt.necy(i))THEN
              write(*,22)i
 22         format('***  WARNING: lifetime of the constellation     ***',
     +           /'***  number ',i2,' shorter than building period ***')
              goto 10
           ENDIF
           goto 10
        ENDIF
c
        lanci=0
        IF(iear.ge.iearco(i))THEN
c  the -1 is necessary since we start the constellation in the year iear
c  so, e.g, if we have a constellation with 73 satellites and we launch
c  16 satellites each year we have   necy=5    (4 years with 16 launches
c  and one with 9 launches); but if we  have iearco = 1994 then
c  iearco + necy = 1999. But we have to launch 16 satellites only from
c  1994 (comprised) to 1997  (i.e. 4 years) and then 9 satellites in
c  1998 (which is  iearco+necy -1 )
c
c  *****  Beginning of the case for the building period
          IF(iear.le.(iearco(i)+necy(i)-1))THEN
             IF(iear.lt.(iearco(i)+necy(i)-1))THEN
c  if some satellite fails in the building period we have to launch
c  a substitute for it
                IF(fail(i).eq.0)THEN
                   nsat(i)=la1(i)*neach(i)
                ELSE
                   nsat(i)=la1(i)*neach(i)+fail(i)
                   icon=1
                ENDIF
                nobr(i)=idnint(obj(i))*la1(i)
c  upper stages
                IF(iear.lt.ieaupc(i))THEN
                   nupc(i)=la1(i)
                ELSE
```

```
                  nupc(i)=0
                ENDIF
                lanci=la1(i)
c  if we have to END the constellation (so we do not need
c  again la1 launches, but the constellation is not yet
c  completed)
c   again we need the  -1 for the same reasons as above
             ELSE IF(iear.eq.iearco(i)+necy(i)-1)THEN
                nsat(i)=nsatt(i)-(la1(i)*neach(i)*(necy(i)-1))
c  launches necessary to END the constellation
                lanci=idnint(dfloat(nsat(i))/dfloat(neach(i)))
                nobr(i)=idnint(obj(i))*lanci
c  upper stages
                IF(iear.lt.ieaupc(i))THEN
                   nupc(i)=lanci
                ELSE
                   nupc(i)=0
                ENDIF
             ENDIF
c  fill the matrix of the objects with the proper mass
c  payloads: since these are the satellites used to build the
c  constellation these are placed on the constellation orbit and
c  are assigned the total constellation lifetime as their lifetime
c  There are also a few failed satellites which are left, uncontrolled,
c  in the same orbit as the constellation
             ifa=1
             DO l=1+nla,nla+nsat(i)
c  the lifetime for the final deorbiting of the satellite is the
c  actual lifetime of the whole constellation
                IF(icon.eq.0)THEN
                   DO j=1,npar_orb
                      cobj(j,l)=eleco(j,i)
                   ENDDO
                   cobj(npar_orb+1,l)=areaco(i)
                   cobj(npar_orb+2,l)=dmasco(i)
                   cobj(npar,l)=idcoco+life(i)+iearco(i)
c  some of the satellites in the building period replace the failed ones
                ELSE IF(icon.eq.1)THEN
                   DO j=1,npar_orb
                      cobj(j,l)=eleco(j,i)
                   ENDDO
                   cobj(npar_orb+1,l)=areaco(i)
```

```
                    cobj(npar_orb+2,l)=dmasco(i)
                    IF(ifa.le.fail(i))THEN
                       ifa=ifa+1
                       cobj(npar,l)=idunco
c  the other satellites in the building period are operative
                    ELSE
                       cobj(npar,l)=idcoco+life(i)+iearco(i)
                    ENDIF
                 ENDIF
              ENDDO
c  END of the case for the building period
           ELSE
c ****  beginning of the case for the maintenance period
              IF(fail(i).eq.0)THEN
                 nobr(i)=idnint(obj(i))*la2(i)
                 nsat(i)=la2(i)*neach(i)
              ELSE
c  if there are failures, substitute the failed satellites
                 nobr(i)=idnint(obj(i))*la2(i)
                 nsat(i)=la2(i)*neach(i)+fail(i)
                 icon=1
              ENDIF
c  upper stages
              IF(iear.lt.ieaupc(i))THEN
                 nupc(i)=la2(i)
              ELSE
                 nupc(i)=0
              ENDIF
              lanci=la2(i)
              ifa=1
              DO l=1+nla,nla+nsat(i)
c  de-orbiting of old satellites
                 IF(ndeo(i).eq.1)THEN
c  the new satellites are immitted on the disposal orbit (i.e. they
c  play the role of the old removed satellites) and have therefore
c  the identity of uncontrolled satellites
c
                    IF(icon.eq.0)THEN
c  no failed satellites to replace; all the launched satellites are
c  going to replace operational ones which are removed into the
c  disposal orbit with the standard procedures applied to all
c  the "normal" satellites
```

```
                        DO ii=1,npar
                           wv0(ii)=wv(ii)
                        ENDDO
c  Distinguish if the de-orbiting policy is the same as the standard
c  satellite or if the constellation has its own disposal orbit
                        IF(deo_pol(i).EQ.0)THEN
                           IF(au_fi.eq.'f')THEN
                              CALL deorb(iear,deflag,reslif,perdeo,
     +                                   deoalt,h1,h2, wv0)
                           ELSE
                              CALL deorb_au(iear,deflag,reslif,perdeo,
     +                                      deoalt,h1,h2,  wv0)
                           ENDIF
                        ELSE
c  the constellation has its own disposal orbit
                           DO j=1,npar_orb
                              wv0(j)=deo_ele(j,i)
                           ENDDO
                        ENDIF
                           DO j=1,npar_orb
                              cobj(j,l)=wv0(j)
                           ENDDO
                           cobj(npar_orb+1,l)=wv0(npar_orb+1)
                           cobj(npar_orb+2,l)=wv0(npar_orb+2)
                           cobj(npar,l)=idunco
                        ELSE IF(icon.eq.1)THEN
c  some of the satellites replace the failed ones therefore
c  we assign them the same orbit as the constellation but with
c  the identity of uncontrolled satellites
                           IF(ifa.le.fail(i))THEN
                              ifa=ifa+1
                              DO j=1,npar_orb
                                 cobj(j,l)=eleco(j,i)
                              ENDDO
                              cobj(npar_orb+1,l)=areaco(i)
                              cobj(npar_orb+2,l)=dmasco(i)
                              cobj(npar,l)=idunco
c  the other are satellites are again injected into the disposal orbit
                           ELSE
                              DO ii=1,npar
                                 wv0(ii)=wv(ii)
                              ENDDO
```

```
c  Distinguish if the de-orbiting policy is the same as the standard
c  satellite or if the constellation has its own disposal orbit
                              IF(deo_pol(i).EQ.0)THEN
                                 IF(au_fi.eq.'f')THEN
                                    CALL deorb(iear,deflag,reslif,perdeo
     +                                       ,deoalt,h1,h2, wv)
                                 ELSE
                                    CALL deorb_au(iear,deflag,reslif,
     +                                         perdeo,deoalt,h1,h2,  wv)
                                 ENDIF
                              ELSE
c  the constellation has its own disposal orbit
                                 DO j=1,npar_orb
                                    wv(j)=deo_ele(j,i)
                                 ENDDO
                              ENDIF
                              DO j=1,npar_orb
                                 cobj(j,l)=wv(j)
                              ENDDO
                              cobj(npar_orb+1,l)=wv(npar_orb+1)
                              cobj(npar_orb+2,l)=wv(npar_orb+2)
                              cobj(npar,l)=idunco
                           ENDIF
                        ENDIF
c  no de-orbiting of old satellites
c  the new ones are injected in the
c  same orbit of the operational constellation and are assigned the
c  uncontrolled flag (i.e. they play the role of the old stranded
c  satellites
                  ELSE
c  here we don't need to distinguish between replacement of failures or
c  replacement of operational since all the replaced satellites stay
c  stranded in the operational orbit
                     DO j=1,npar_orb
                        cobj(j,l)=eleco(j,i)
                     ENDDO
                     cobj(npar_orb+1,l)=areaco(i)
                     cobj(npar_orb+2,l)=dmasco(i)
                     cobj(npar,l)=idunco
                  ENDIF
               ENDDO
c  END of the case for the maintenance period
```

```
            ENDIF
c  fill the matrix of the objects with the proper mass
c  upper stages
            DO l=1+nla+nsat(i),nla+nsat(i)+nupc(i)
               DO j=1,npar_orb
                  cobj(j,l)=eleusco(j,i)
               ENDDO
               cobj(npar_orb+1,l)=areuco(i)
               cobj(npar_orb+2,l)=dmauco(i)
               cobj(npar,l)=idups
            ENDDO
c  mission related objects (they are supposed to be left in the
c  same orbit as the upper stage)
            DO l=1+nla+nsat(i)+nupc(i),nla+nsat(i)+nupc(i)+
     $                                    nobr(i)
               DO j=1,npar_orb
                  cobj(j,l)=eleusco(j,i)
               ENDDO
               cobj(npar_orb+1,l)=a_m(dmarec,ia_m)
               cobj(npar_orb+2,l)=dmarec
               cobj(npar,l)=idmrd
            ENDDO
c  total number of objects launched in the year, due to constellations
            nla=nla+nsat(i)+nupc(i)+nobr(i)
            nlanci=nlanci+lanci
         ENDIF
 10   CONTINUE
      close(4)
      RETURN
      END
c  ===================================================================
c   SUBROUTINE STRUCT
c   Version  3.0   of 1/7/2003
c
c  PURPOSE
c  Simulates the building of large structures (e.g. space stations)
c
c  INPUT ARGUMENTS
c      iear       year of the simulation
c
c  OUTPUT ARGUMENTS
c      nstr       total number of objects injected in orbit
```

```
c                in the year  iear   due to large structures
c      nla       number of objects launched in the year, due to structures
c      lanci     number of servicing missions per year
c      sobj      matrix with the objects launched
c
c  CALL SUBROUTINES
c     zero, reastr
c
c  INCLUDE FILES
c    parlau.h, stastr.h
c
c  REFERENCE
c     Study Notes of the WP2200
c
c  ANALYSIS
c   A. Rossi
c  PROGRAMMER
c   A. Rossi
c  PROGRAM MODIFICATIONS
c   none
c  COMMENTS
c   UNIX version
c
c  ================================================================
      SUBROUTINE struct(iear, nstr,nla,lanci,sobj)
c
      IMPLICIT none
      INCLUDE "parlau.h"
      INCLUDE "stastr.h"
      INCLUDE 'parbin.h'
c  INPUT
      INTEGER iear
c  OUTPUT
      INTEGER nstr,nla,lanci
      DOUBLE PRECISION sobj(npar,nmax)
c
      INTEGER iearst(nstrx),lastr(nstrx),nplast(nstrx),nlas(nstrx)
      INTEGER ifir(nstrx),ll,i,ia_m,n,j
      DOUBLE PRECISION elestr(npar_orb,nstrx)
      DOUBLE PRECISION objstr(nstrx),dmassa,a_m
      LOGICAL first
c
```

```
      DATA first/.true./
      INCLUDE "idcode.h"
c  LINUX VERSION
      save
c
      lanci=0
      nla=0
c  "initial" mass of the structure in kg (100 tons)
      dmassa=1.d5
      CALL zero(nlas,nstrx)
c
      IF(first) THEN
c  file with the characteristic of the structure (unit 4)
c  read the file launch.in
c  # of structures,year, # of mission/year
c  object realeased per mission, mass added to the structures per
c  mission, a, e, i
         CALL reastr(4,nstr,iearst,nplast,lastr,objstr,elestr)
         close(4)
         DO 100 ll=1,nstrx
            ifir(ll)=1
 100     CONTINUE
         first=.false.
      ENDIF
c     check on the years
      DO 10 i=1,nstr
         IF(iear.ge.iearst(i).and.iear.le.iearst(i)+nplast(i))THEN
c  if it is the first time a structure is considered, we
c  assign it a mass of 100 tons; afterwards the mass is no more
c  changed and only the mission related debris are injected in orbit
            IF(ifir(i).eq.1)THEN
               DO j=1,npar_orb
                  sobj(j,nla+1)=elestr(j,i)
               ENDDO
               sobj(npar_orb+1,nla+1)=a_m(dmassa,ia_m)
               sobj(npar_orb+2,nla+1)=dmassa
               sobj(npar,nla+1)=idspa+nplast(i)+iear
            ENDIF
c  total number of objects injected in orbit (not connected
c  to the structure)
            nlas(i)=lastr(i)*idnint(objstr(i))
c  add the mission related objects
```

```
c  in the launches related to structure we suppose that there
c  are no payloads (not connected to the structure) and no
c  upper stages (all the missions are performed with a shuttle)
           DO 20 n=1,nlas(i)
              DO j=1,npar_orb
                 sobj(j,nla+ifir(i)+n)=elestr(j,i)
              ENDDO
              sobj(npar_orb+1,nla+ifir(i)+n)=a_m(dmares,ia_m)
              sobj(npar_orb+2,nla+ifir(i)+n)=dmares
              sobj(npar,nla+ifir(i)+n)=idmrd
 20         CONTINUE
            lanci=lanci+lastr(i)
            nla=nla+nlas(i)+ifir(i)
            ifir(i)=0
         ENDIF
c  total number of objects launched in the year, due to structures
 10   CONTINUE
      RETURN
      END
c  ===================================================================
c     SUBROUTINE RETRIE
c  Version  3.0   of  22/5/2002
c
c  PURPOSE
c  Simulates retrievals
c
c  INPUT ARGUMENTS
c     iear   year
c
c  OUTPUT ARGUMENTS
c
c  CALL SUBROUTINES
c    rearet.f ,elimmh.f
c  INCLUDE FILES
c    parlau.h, parbin.h, paralt.h
c
c  REFERENCE
c     Study Notes of the WP2200
c
c  ANALYSIS
c   A. Rossi
c  PROGRAMMER
```

```
c   A. Rossi
c  PROGRAM MODIFICATIONS
c   none
c  COMMENTS
c   UNIX version
c  ================================================================
      SUBROUTINE retrie(iear)
c
      IMPLICIT none
c  INPUT
      INTEGER iear
c
      INCLUDE "parlau.h"
      INCLUDE "parbin.h"
      INCLUDE "paralt.h"
      DOUBLE PRECISION smar(nclasx),ecr(nclasx),dinr(nclasx)
      DOUBLE PRECISION dmar(nclasx),drate(nclasx),dnel(nhshx)
      DOUBLE PRECISION elim(nhshx,nbinm)
      INTEGER inir(nclasx),iendr(nclasx),i,ielh,nretr,nmr
      LOGICAL first
c      character name*6,type*3
c  matrix with the number of objects to be subtracted from each
c  altitude shell for each mass bin (mass bins x altitude shells)
      COMMON/elimin/elim
      data first/.true./
c  LINUX VERSION
      save
c
      IF(first)THEN
c  file with the retrieval rates (unit 57)
c  read the file retrie.dat
         CALL rearet(57, nretr,smar,ecr,dinr,dmar,drate,inir,iENDr)
         first=.false.
      ENDIF
      DO i=1,nretr
         IF(iear.ge.inir(i).and.iear.le.iendr(i))THEN
            CALL elimmh(dmar(i),smar(i),ecr(i), nmr,dnel)
            DO ielh=1,nhshx
               elim(ielh,nmr)=elim(ielh,nmr)+dnel(ielh)*drate(i)
            ENDDO
         ENDIF
      ENDDO
```

```
      RETURN
      END
c  ====================================================================
c   SUBROUTINE ASSCLA
c   Version  3.0  of 30/5/2003
c
c  PURPOSE
c   Assign to each object injected in orbit the semimajor axis,
c   the eccentricity, the inclination, the area and the mass
c
c  INPUT ARGUMENTS
c     iyear   year of the simulation
c     iref    reference year for the launches
c     nlo     number of launches
c     nclas   number of classes of the objects launched
c     clas    area, mass, orbital elements elements of the classes
c             clas(1,i)=a
c             clas(2,i)=e
c             clas(3,1)=i ,
c             clas(4,i)=area [m^2]
c             clas(5,i)=mass [kg]
c             clas (6,i)=identification flag+lifetime [in years]
c     dper    percentages of the launches for each class
c     dtr     trend for the masses (e.g. 0.01 means 1 % growth in
c             the masses)
c     paylay  number of payload for each class
c     iup     flag for the upper stages:
c                           0 = no upper stage left in orbit
c                           1 = one upper stage left in orbit
c                           2 = two upper stages left in orbit
c     claus1  matrix with the elements and mass of the first upper stage
c     msolid1 mass of the solid propellant of upper stage 1
c     a_p1    apogee or perigee motor (upper stage 1)
c     limu1   time after which the first upper stage is no more left
c             in orbit
c     claus2  matrix with the elements and mass of the second upper stage
c     msolid2 mass of solid propellant of upper stage 2
c     a_p2    apogee or perigee motor (upper stage 2)
c     limu2   time after which the second upper stage is no more left
c             in orbit
c     relo    number of mission related objects left in orbit for each
c             class
```

```
c     limr    time after which the mission related objects are no
c             more left in orbit
c     aml     lower limits of the mass bins (kg)
c     amh     upper limits of the mass bins (kg)
c
c  OUTPUT ARGUMENTS
c          dlobj   matrix with the elements of the objects launched
c          n0      total number of objects distributed in orbit
c          nactla  actual number of launches performed (taken into
c                  account the Poisson extraction for small percentages)
c          bislag  slag/liner particles particles matrix
c          nslag   number of slag/liner particles in each mass bins
c
c  CALL SUBROUTINES
c    xtrobj.f
c
c  INCLUDE FILES
c     parlau.h, parexp.h, stalau.h, idcode.h, parsla.h
c
c  REFERENCE
c     Study Notes of the WP2200
c
c  ANALYSIS
c   A. Rossi
c  PROGRAMMER
c   A. Rossi
c  PROGRAM MODIFICATIONS
c   none
c  COMMENTS
c   UNIX version
c
c  ================================================================
      SUBROUTINE asscla(iyear,iref,nlo,nclas,clas,dper,dtr,paylau,
     +                  iup,claus1,msolid1,a_p1,limu1,claus2,
     +                  msolid2,a_p2,limu2,relo,limr,aml,amh, dlobj,n0,
     +                  nactla,bislag,nslag)
c
      IMPLICIT none
      INCLUDE "parlau.h"
      INCLUDE 'parbin.h'
      INCLUDE 'parsla.h'
c  parexp is include since we need the quantity nbin
```

```
      INCLUDE "parexp.h"
c  INPUT
      INTEGER iyear,iref,nlo,nclas,iup(nclas),limu1(nclas),limu2(nclas)
      INTEGER limr(nclas)
      DOUBLE PRECISION clas(npar,nclas),dper(nclas),dtr(nclas)
      DOUBLE PRECISION paylau(nclas),claus1(npar,nclas)
      DOUBLE PRECISION claus2(npar,nclas),relo(nclas)
      DOUBLE PRECISION aml(nbinm),amh(nbinm)
      DOUBLE PRECISION msolid1(nclas),msolid2(nclas)
      CHARACTER*1 a_p1(nclas),a_p2(nclas)
c  OUTPUT
      DOUBLE PRECISION dlobj(npar,nmax),bislag(npar,nsabsl,nbinm)
      INTEGER n0,nactla,nslag(nbinm)
c
c  number of launches for each class
      INTEGER nojcla(nclasx)
      DOUBLE PRECISION p(nbin)
      DOUBLE PRECISION delta,dnojcla,dn,dmapay,delare,arepay,dmas1,a_m
      DOUBLE PRECISION ares1,dmas2,ares2,v,v1,di,e,a,disu,esu,asu
c  SLAG/LINER
      DOUBLE PRECISION elesla(6)
      INTEGER n_sla_rel
c
      INTEGER ia_m,i,n,m,npa,nre,nup,nt
      COMMON/areama/ia_m
c
      INCLUDE "stalau.h"
c  Identification codes for different kind of objects
      INCLUDE "idcode.h"
c  LINUX VERSION
      save
c
      n0=0
c  years from the reference year
      delta=dfloat(iyear-iref)
c  actual number of launches
      nactla=0
c Slag/Liner inizialization
      nslag=0
      bislag=0.d0
c  Number of slag/liner releases
      n_sla_rel=0
```

```
c  calculates the launches for each class
      DO 10 i=1,nclas
c  number of launches in the class i
c         nojcla(i)=idnint(dfloat(nlo)*dper(i)/100.d0)
c  if the number of launches comes out to be a small
c  rational number then we apply the Poisson extractor
          dnojcla=dfloat(nlo)*dper(i)/100.d0
          IF(dnojcla.lt.5.d-1)THEN
             CALL poispr(dnojcla, dn)
             nojcla(i)=idnint(dn)
          ELSE
             nojcla(i)=idnint(dnojcla)
          ENDIF
          nactla=nactla+nojcla(i)
c  mass of the class
         dmapay=clas(npar_orb+2,i)*(1.d0+dtr(i))**delta
c  Area of the class
c  The area to be added correspondingly to the growth in mass
c  is calculated differentiating tha Kessler's (1989) relation
         delare=1.13d0*63.8d0*(dmapay-clas(npar_orb+2,i))*
     +          clas(npar_orb+2,i)**0.13d0
         arepay=clas(npar_orb+1,i)+delare
c  number of payload
         npa=idnint(paylau(i))
c  number of mission related objects
         IF(iyear.lt.limr(i))THEN
            nre=idnint(relo(i))
         ELSE
            nre=0
         ENDIF
c  upper stages mass assignement and counting
         IF(iyear.lt.limu1(i).and.iup(i).ge.1)THEN
            nup=1
            dmas1=claus1(npar_orb+2,i)*(1.d0+dtr(i))**delta
            delare=1.13d0*63.8d0*(dmas1-claus1(npar_orb+2,i))
     +             *claus1(npar_orb+2,i)**0.13d0
            ares1=claus1(npar_orb+1,i)+delare
         ELSE IF(iup(i).eq.2.and.iyear.lt.limu2(i))THEN
            nup=2
            dmas2=claus2(npar_orb+2,i)*(1.d0+dtr(i))**delta
            delare=1.13d0*63.8d0*(dmas2-claus2(npar_orb+2,i))
     +             *claus2(npar_orb+2,i)**0.13d0
```

```
            ares2=claus2(npar_orb+1,i)+delare
          ELSE
            nup=0
          ENDIF
c  total number of objects injected in orbit in this class
          nt=npa+nup+nre
c  loop on the number of launches for each class
          DO 20 n=1,nojcla(i)
            IF(clas(1,i).ne.0.d0)THEN
c  fake values used to call xtrobj (not used in this case)
              v=10.d0
              v1=5.d0
c  choose i,e,a  in an interval centered on the classes values
              CALL xtrobj(clas(3,i)+dei,clas(3,i)-dei,
     +         v,v1,clas(2,i)+dee,clas(2,i)-dee,clas(1,i)+dea,
     +         clas(1,i)-dea, nbin, di,v,e,a, p)
            ELSE
              a=0.d0
              e=0.d0
              di=0.d0
            ENDIF
c  payloads
            DO m=1,npa
              dlobj(1,n0+m)=dabs(a)
              dlobj(2,n0+m)=dabs(e)
              dlobj(3,n0+m)=dabs(di)
              dlobj(npar_orb+1,n0+m)=arepay
              dlobj(npar_orb+2,n0+m)=dmapay
              dlobj(npar,n0+m)=clas(npar,i)+iyear
            ENDDO
c  upper stages
            IF(nup.EQ.1)THEN
c  first upper stage
c  choose i,e,a  in an interval centered on the classes values
              CALL xtrobj(claus1(3,i)+dei,claus1(3,i)-dei,
     +         v,v1,claus1(2,i)+dee,claus1(2,i)-dee,claus1(1,i)+dea,
     +         claus1(1,i)-dea, nbin, disu,v,esu,asu, p)
              dlobj(1,n0+npa+nup)=dabs(asu)
              dlobj(2,n0+npa+nup)=dabs(esu)
              dlobj(3,n0+npa+nup)=dabs(disu)
              dlobj(npar_orb+1,n0+npa+nup)=ares1
              dlobj(npar_orb+2,n0+npa+nup)=dmas1
```

```
               dlobj(npar,n0+npa+nup)=idups
c SLAG/LINER release (assume 5 orbital elements are read from the
c lclass file)
               IF(msolid1(i).GT.0.d0)THEN
                  elesla=0.d0
                  DO m=1,npar_orb
                     elesla(m)=claus1(m,i)
                  ENDDO
                  elesla(6)=0.d0
c  SLAG (50 % of the mass)
                  n_sla_rel=n_sla_rel+1
                  CALL slagen(msolid1(i)/2.d0,elesla,a_p1(i),aml,amh,
     +                        roslag, bislag,nslag)
c  LINER (50 % of the mass)
                  CALL slagen(msolid1(i)/2.d0,elesla,a_p1(i),aml,amh,
     +                        rolin, bislag,nslag)
               ENDIF
            ENDIF
            IF(nup.EQ.2)THEN
c  second upper stage
c  choose i,e,a  in an interval centered on the classes values
               CALL xtrobj(claus2(3,i)+dei,claus2(3,i)-dei,
     +         v,v1,claus2(2,i)+dee,claus2(2,i)-dee,claus2(1,i)+dea,
     +         claus2(1,i)-dea, nbin, disu,v,esu,asu, p)
               dlobj(1,n0+npa+nup)=dabs(asu)
               dlobj(2,n0+npa+nup)=dabs(esu)
               dlobj(3,n0+npa+nup)=dabs(disu)
               dlobj(npar_orb+1,n0+npa+nup)=ares2
               dlobj(npar_orb+2,n0+npa+nup)=dmas2
               dlobj(npar,n0+npa+nup)=idups
c SLAG/LINER release (assume 5 orbital elements are read from the
c lclass file)
               IF(msolid2(i).GT.0.d0)THEN
                  elesla=0.d0
                  DO m=1,npar_orb
                     elesla(m)=claus2(m,i)
                  ENDDO
                  elesla(6)=0.d0
c  SLAG (50 % of the mass)
                  n_sla_rel=n_sla_rel+1
                  CALL slagen(msolid2(i)/2.d0,elesla,a_p1(i),aml,amh,
     +                        roslag, bislag,nslag)
```

```
c  LINER (50 % of the mass)
                  CALL slagen(msolid2(i)/2.d0,elesla,a_p1(i),aml,amh,
     +                        rolin, bislag,nslag)
               ENDIF
            ENDIF
c  mission related objects
            DO m=1,nre
               dlobj(1,n0+npa+nup+m)=dabs(a)
               dlobj(2,n0+npa+nup+m)=dabs(e)
               dlobj(3,n0+npa+nup+m)=dabs(di)
               dlobj(npar_orb+1,n0+npa+nup+m)=a_m(dmasre,ia_m)
               dlobj(npar_orb+2,n0+npa+nup+m)=dmasre
               dlobj(npar,n0+npa+nup+m)=idmrd
            ENDDO
            n0=n0+nt
c  END of loop on the number of launches in each class
 20      CONTINUE
c  END of loop on the classes
 10   CONTINUE
      IF(n_sla_rel.GT.0)THEN
         WRITE(*,*)
         WRITE(*,*)'************************************************'
         WRITE(*,*)'****** ',n_sla_rel,' SLAG/LINER RELEASES *******'
         WRITE(*,*)'************************************************'
         WRITE(*,*)
      ENDIF
      RETURN
      END
```

A.6 HANADA.H

The header file *hanada.h* contains the impact velocity, in km/s, below which the Hanada model for low velocity collisions is used.

```
c  Limit for low velocity impacts (i.e. impact velocity
c  below which Hanada's model is used) [in km/sec]
      DOUBLE PRECISION vlim_h
      DATA vlim_h/5.d0/
```

A.7 COLOBJ.H

```
c  Avearge mass in LEO, MEO and GEO for every mass bin, in
c  the MASTER 2001 model
      DOUBLE PRECISION av_leo(nbinm),av_meo(nbinm),av_geo(nbinm)
c  Standard deviation of the mass in LEO, MEO and GEO for every
c  mass bin, in the MASTER 2001 model
      DOUBLE PRECISION std_leo(nbinm),std_meo(nbinm),std_geo(nbinm)
c
      DATA av_leo/2.068e-06,2.463e-05,2.685e-04,0.0028d0,0.0260d0,
     +            0.299d0,3.482d0,39.d0,479.d0,2148.d0/
      DATA av_meo/2.068e-06,2.463e-05,2.685e-04,0.0028d0,0.0260d0,
     +            0.299d0,3.482d0,55.d0,607.d0,1741.d0/
      DATA av_geo/2.068e-06,2.463e-05,2.685e-04,0.0028d0,0.0260d0,
     +            0.299d0,3.482d0,37.d0,484.d0,2076.d0/
      DATA std_leo/2.48e-06,2.49e-05,2.49e-04,0.0024d0,0.0212d0,
     +            0.2259d0,2.1812d0,19.d0,270.d0,1743.d0/
      DATA std_meo/2.48e-06,2.49e-05,2.49e-04,0.0024d0,0.0212d0,
     +      0.2259d0,2.1812d0,25.d0,332.d0,1003.d0/
      DATA std_geo/2.48e-06,2.49e-05,2.49e-04,0.0024d0,0.0212d0,
     +      0.2259d0,2.1812d0,23.d0,234.d0,775.d0/
```

A.8 DATDCP.H

```
c  ===========================================================
c   datdcp.h
c  Data used in the Debris Cloud Propagator
c  CD used for the drag calculations
      DOUBLE PRECISION cd,dlambd,rmi,rmxdcp,epsi,aic,dminfl
      data cd/2.2d0/
c  Multiplicative factor for the drag calculations
      data dlambd/1.2d0/
c  Lowest altitude permitted for an orbit (km)
      data rmi/6508.d0/
c  Upper limit of the atmosphere (km)
      data rmxdcp/7578.14d0/
c  Distance between apogee and perigee below which the orbit
c  is considered "near circular" so that the apogee height
c  start to decrease as well (km)
      data epsi/50.d0/
c  distance from the last minimum of the solar activity in years
c     data aic/7.2d0/
```

```
c  Year of the last minimum of the solar flux
      data dminfl/1996.4d0/
c  ============================================================
```

A.9 DATEXP.H

```
c  ============================================================
c  datexp.h
c
c  AMMIN=the treshold value under which masses are considered not
c        one by one but as a distribution (kg);
c  If this value has to be changed the value of the parameter
c  nsmall in parbin.h has to be changed consistently
c  (i.e. ammin must be the upper limit of the  nsmall bin)
      DOUBLE PRECISION ammin,ammin0
      data ammin/1.d-2/
c  AMMIN0=lower cutoff for the masses (Kg);
      data ammin0/1.d-6/
c  ============================================================
```

A.10 DIRDEH.H

```
c  ====================================================================
c  Directory containing the files with the densities of the
c  historical population
      dirfih='DENSHIS/dens.dat'
```

A.11 DIRDEH_CD.H

```
c  ====================================================================
c  Directory containing the files with the densities of the
c  historical population
      dirfih_cd='/mnt/cdrom/DENSHIS/dens.dat'
```

A.12 DIRDER.H

```
 ====================================================================
c  Directory containing the files with the densities of the
c  historical population
      dirfir='DENSHIS/ADDITIONAL/dens.dat'
```

A.13 DIRDER_CD.H

```
c  ===================================================================
c  Directory containing the files with the densities of the
c  historical population
      dirfir_cd='/mnt/cdrom/DENSHIS/ADDITIONAL/dens.dat'
```

A.14 PARBIN.H

Header file with some of the fundamental settings of SDM, e.g. number of mass bin, dimensioning of the main matrix.

```
c  ====================================================
c parbin.h
c
      INTEGER nbinm,npar,nidc,ndp,npar_orb
c  number of mass bins
      parameter(nbinm=10)
c  Number of values identifing each object:
c   bin(1,i,j)=a [km]
c   bin(2,i,j)=e
c   bin(3,1,j)=i [degrees]
c     bin(4,i,j)=argument of node [degrees]
c     bin(5,i,j)=argument of perigee [degrees]
c     bin(6,1,j)=mean anomaly [degrees]
c   bin(7,i,j)=area [m^2]
c   bin(8,i,j)=mass [kg]
c   bin(9,i,j)=identification flag+lifetime
c   Orbital parameters used (6 to allow the use of FOP)
      parameter(npar_orb=6)
c  Total number of orbital parameter
      parameter(npar=npar_orb+3)
c  Number of classes of objects whose identification codes
c  are assigned in     idcode.h
      parameter(nidc=14)
c  Parameter needed for the dimensioning of the matrix with
c  the percentages of objects
      parameter(ndp=nbinm+1)
c  ========================================================
```

A.15 PARPRO.H

```
c  ============================================================
c   parpro.h
c  Limiting eccentricity above which we use FOP instead of
c  DCP if iprop=1
      DOUBLE PRECISION limec
      PARAMETER(limec=0.14d0)
c  ============================================================
```

A.16 PROP_LIM.H

The header file *prop_lim.h* contains the orbital parameters (a and e) discriminating the orbits that are propagated, in the pre-processing code, with DCP or FOP.

```
c  Thresholds to switch from DCP to FOP:
c  if a > 8378 km or e > 0.1 ====> use FOP
      DOUBLE PRECISION lima,lime
      DATA lima/8378.14d0/
      DATA lime/0.1d0/
```

B Appendix: Macros

B.1 CREDEN

Following is the text of the macro *creden* used to run the density generation program:

```
#!      /bin/sh

#  Shell script to run CREDEN
#
PAT_SDM=/home/rossi/debris
#
echo   '***  Creating historical densitites files'
echo ''
#
cd $PAT_SDM/SDM/DENSHIS/WORK
echo ''
echo 'Enter the name of the input file with the elements of the'
echo 'population to be read'
echo 'WARNING: the full path of the file has to be given'
echo '(e.g. /home/SDM/DENSHIS/WORK/MASTER/elements.dat)'
echo ''
read dirinp
echo ''
#
echo 'Enter the name of the directory in which to store
      the density files'
echo 'Possible choiches:
echo '           1) DENSHIS  (if it is the standard population)'
echo '           2) DENSHIS/ADDITIONAL'
echo '                       (if it is an additional population)'
echo 'Enter choice 1 or 2'
echo ''
read nansw
if (test "$nansw" = "1")
        then    direcout='DENSHIS'
        elif (test "$nansw" = "2")
        then    direcout='DENSHIS/ADDITIONAL'
fi
echo ''
echo $PAT_SDM/SDM/$direcout
#
ln $dirinp fort.1
```

```
ln $PAT_SDM/SDM/DENSHIS/WORK/creden.inp fort.2
ln $PAT_SDM/SDM/DENSHIS/WORK/densinp.dat fort.4
#
time $PAT_SDM/SDM/DENSHIS/WORK/creden.x
#
\mv dens.* $PAT_SDM/SDM/$direcout
\rm fort.1
\rm fort.2
\rm fort.4
#
echo ''
echo ' *******  Program terminated normally  *******'
echo ' *******  Historical densities stored in ******'
echo ' *******  '$PAT_SDM'/SDM/'$direcout' *******'
```

B.2 SDM

Following is the text of the macro *sdm* used to run the main SDM program:

```
#!      /bin/sh

PAT_SDM=/home/rossi/debris

# Read historical densities from hard disk
echo '0'> fort.17

#  Shell script to run SDM
#  It run SDM in the MAIN directory (e.g. debris) and
#  read the input files from the directory specified first and
#  stores the output files in the directory specified as second.
#  Launch the script with a command like
#        sdm2 input-directoryname output-directoryname
#  NB. The input-directoryname output-directoryname can be the
#      same directory
#
usage () {
        echo "Usage: sdm input-directory output-directory"
        exit
        }

usage1 () {
        echo 'Input directory not existent'
        exit
```

```
        }

if [ $# -lt 2 ]; then
        usage
fi

if [ ! -d "$1" ]; then
        usage1
fi


mkdir $2

#
echo   '***  Input files read from the directory' $1
echo   '***  Creating directory' $2
echo   '***  All the output files are stored in' $2
echo ''
echo ''
#
ln $PAT_SDM/SDM/$1/sdm.dat fort.1
ln $PAT_SDM/SDM/$1/seed.dat fort.2
ln $PAT_SDM/SDM/$1/seed.old fort.3
ln $PAT_SDM/SDM/$1/launch.in fort.4
#ln $PAT_SDM/SDM/$1/lclass.dat fort.8
\cp -p $PAT_SDM/SDM/$1/lclass_* $PAT_SDM/SDM
ln $PAT_SDM/SDM/$1/expinp.dat fort.9
ln $PAT_SDM/SDM/$1/collis.dat fort.12
ln $PAT_SDM/SDM/$1/retrie.dat fort.57
ln $PAT_SDM/SDM/$1/rorsat.dat fort.19
ln $PAT_SDM/SDM/$1/snapsyears.dat fort.10
#
time ${PAT_SDM}/SDM/sdm.x
# Copy input files for critical density calculations
\cp $PAT_SDM/SDM/$1/collis.dat $PAT_SDM/SDM/$2
\cp $PAT_SDM/SDM/$1/decay.tab $PAT_SDM/SDM/$2
\cp $PAT_SDM/SDM/$1/meanum.dat $PAT_SDM/SDM/$2
# move running population files to the proper directory
\mv running.???? $PAT_SDM/SDM/$2
#
\mv fort.31 $PAT_SDM/SDM/$2/densmas.out
\mv fort.32 $PAT_SDM/SDM/$2/densdia.out
```

```
#
\mv fort.41 $PAT_SDM/SDM/$2/number.dat
\mv fort.42 $PAT_SDM/SDM/$2/numdia.dat
\mv fort.43 $PAT_SDM/SDM/$2/percen.dat
\mv fort.44 $PAT_SDM/SDM/$2/numberru.dat
\mv fort.45 $PAT_SDM/SDM/$2/numdiaru.dat
#
\mv fort.48 $PAT_SDM/SDM/$2/warn.out
\mv fort.47 $PAT_SDM/SDM/$2/launch.out
\mv fort.53 $PAT_SDM/SDM/$2/explod.ele
\mv fort.54 $PAT_SDM/SDM/$2/collid.ele
\mv fort.58 $PAT_SDM/SDM/$2/rorsat.ele
\mv fort.55 $PAT_SDM/SDM/$2/flux.out
\mv fort.56 $PAT_SDM/SDM/$2/procol.out
\mv fort.95 $PAT_SDM/SDM/$2/reentry_check.dat
\mv fort.86 $PAT_SDM/SDM/$2/deltav.dat
#\mv fort.98 $PAT_SDM/SDM/$2/running_pop.dat
# copy the file sdm.dat in the directory chosen because
# the postprocessor needs to read from this file the
# flag for the area to mass relation
\cp $1/sdm.dat $2
# compress the running population snapshots files
gzip -f $PAT_SDM/SDM/$2/running*
#
\rm fort.1
\rm fort.2
\rm fort.3
\rm fort.4
#\rm fort.8
\rm lclass_*.dat
\rm fort.9
\rm fort.10
\rm fort.12
\rm fort.17
\rm fort.57
\rm fort.19
```

B.3 CD_SDM

```
#!      /bin/sh

# Mount the cd rom drive
```

```
mount /mnt/cdrom
echo '1'> fort.17
#

PAT_SDM=/home/rossi/debris

#  Shell script to run SDM
#  It run SDM in the MAIN directory (e.g. debris) and
#  read the input files from the directory specified first and
#  stores the output files in the directory specified as second.
#  Launch the script with a command like
#       sdm2 input-directoryname output-directoryname
#  NB. The input-directoryname output-directoryname can be the
#      same directory
#
usage () {
        echo "Usage: sdm input-directory output-directory"
        exit
        }

usage1 () {
        echo 'Input directory not existent'
        exit
        }

if [ $# -lt 2 ]; then
        usage
fi

if [ ! -d "$1" ]; then
        usage1
fi


mkdir $2

#
echo   '***  Input files read from the directory' $1
echo   '***  Creating directory' $2
echo   '***  All the output files are stored in' $2
echo ''
echo ''
```

```
#
ln -f $PAT_SDM/SDM/$1/sdm.dat fort.1
ln -f $PAT_SDM/SDM/$1/seed.dat fort.2
ln -f $PAT_SDM/SDM/$1/seed.old fort.3
ln -f $PAT_SDM/SDM/$1/launch.in fort.4
#ln -f $PAT_SDM/SDM/$1/lclass.dat fort.8
\cp -p $PAT_SDM/SDM/$1/lclass_*.dat $PAT_SDM/SDM
ln -f $PAT_SDM/SDM/$1/expinp.dat fort.9
ln -f $PAT_SDM/SDM/$1/collis.dat fort.12
ln -f $PAT_SDM/SDM/$1/retrie.dat fort.57
ln -f $PAT_SDM/SDM/$1/rorsat.dat fort.19
ln -f $PAT_SDM/SDM/$1/snapsyears.dat fort.10
ln -f $PAT_SDM/SDM/$1/input.fop fort.15
#
time ${PAT_SDM}/SDM/sdm.x
#
# move running population files to the proper directory
\mv running.???? $PAT_SDM/SDM/$2
#
\mv fort.31 $PAT_SDM/SDM/$2/densmas.out
\mv fort.32 $PAT_SDM/SDM/$2/densdia.out
#
\mv fort.41 $PAT_SDM/SDM/$2/number.dat
\mv fort.42 $PAT_SDM/SDM/$2/numdia.dat
\mv fort.43 $PAT_SDM/SDM/$2/percen.dat
\mv fort.44 $PAT_SDM/SDM/$2/numberru.dat
\mv fort.45 $PAT_SDM/SDM/$2/numdiaru.dat
#
if [ -r "fort.48" ]; then
        \mv fort.48 $PAT_SDM/SDM/$2/warn.out
fi
\mv fort.47 $PAT_SDM/SDM/$2/launch.out
\mv fort.53 $PAT_SDM/SDM/$2/explod.ele
\mv fort.54 $PAT_SDM/SDM/$2/collid.ele
if [ -r "fort.58" ]; then
        \mv fort.58 $PAT_SDM/SDM/$2/rorsat.ele
fi
\mv fort.55 $PAT_SDM/SDM/$2/flux.out
\mv fort.56 $PAT_SDM/SDM/$2/procol.out
if [ -r "fort.95" ]; then
        \mv fort.95 $PAT_SDM/SDM/$2/reentry_check.dat
fi
```

```
if [ -r "fort.86" ]; then
        \mv fort.86 $PAT_SDM/SDM/$2/deltav.dat
fi
#\mv fort.98 $PAT_SDM/SDM/$2/running_pop.dat
# copy the file sdm.dat in the directory chosen because
# the postprocessor needs to read from this file the
# flag for the area to mass relation
\cp $1/sdm.dat $2
# compress the running population snapshots files
gzip -f $PAT_SDM/SDM/$2/running*
#
\rm fort.1
\rm fort.2
\rm fort.3
\rm fort.4
#\rm fort.8
\rm lclass_*.dat
\rm fort.9
\rm fort.10
\rm fort.12
\rm fort.17
\rm fort.57
\rm fort.19 fort.15
```

C Appendix: Input files

C.1 EXPINP.DAT

This is the file with the definition of the explosion scenario.

```
#                          Classes of explosions
#  a        e       i      Arg.Per.  mass  factor  exp/y  h/l sc_rb   Interval
 6990.0  0.0089   98.00       0.     900.0  1.0000  0.174  0    r    2000  2010
21578.0  0.6859    7.20       0.    1600.0  0.3831  0.522  0    r    1990  2015
 7114.0  0.0014   98.50       0.    1000.0  0.1741  0.174  0    r    1995  2010
 6790.0  0.0004   65.00       0.    3000.0  0.0882  0.348  1    s    1990  2010
 7825.0  0.0047   82.60       0.    1360.0  0.7493  0.174  0    r    1990  2020
 6856.0  0.0011   73.50       0.    1360.0  0.7493  0.174  0    r    1990  2020
12000.0  0.4593   46.50       0.      55.0  0.6233  0.348  0    r    1990  2030
16000.0  0.5823   52.00       0.      55.0  0.1614  0.174  0    r    1990  2030
16000.0  0.5823   65.00       0.      55.0  0.1614  0.348  0    r    1990  2030
```

The explosion in each line refers to the following spacecraft type:

1. India PSLV 4th Stage
2. ESA Ariane 2/3/4 3rd Stage
3. China CZ-4 3rd Stage
4. Russia EORSAT Elint Satellite
5. Ucraina Tsyklon 3 3rd Stage
6. Ucraina Tsyklon 3 3rd Stage
7. Russia Proton SOZ Ullage Motor
8. Russia Proton SOZ Ullage Motor
9. Russia Proton SOZ Ullage Motor

C.2 LAUNCH.IN

This is the file with the definition of the launch traffic.

```
2004 ; iref    reference year for the number of launches
56   ; nlaunch number of routine BAU launches/year
;trends for the future years
0  ;  0 = single lclass file --- 1 = one lclass file per decade
; interval        trend
```

```
  2004 2104        0.00
111111111  1111111 1111111
CONSTELLATIONS
6    ; nconst  number of constellations to be launched
; (IRIDIUM)
1997 ; iear    year of first launch (IRIDIUM)
20   ; life    planned lifetime of the constellation (years)
78   ; nsat    number of satellites in the constellation
5    ; neach   number of satellites for each launch
9    ; la1     launches/year in the period of construction of the constellation
1    ; la2     launches/year to maintain the constellation
7    ; areaco  area (in m^2) of each satellite
575. ; dmasco  mass (in kg) of each satellite
7158   0.00   86.4  0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap
1    ; ndeo    deorbiting of old satellites? (1=yes, 0=no)
0    ; deo_pol deorbiting policy (0 = standard;  1 = non standard)
6758   0.00   86.4  0.0  ; non-standard deorbiting orbit (if deo_pol=1)
10.  0.001  0.1  0.1 ; spread around the nominal storage orbit
0.2  ; fail    failure rate [number of satellites/year]
1997 ;  upper stage left in orbit and year of change policy
20.   ; areuco  area (in m^2) of the upper stage for the constellation
1000. ; dmauco  mass (in kg) of the upper stage for the constellation
6824.   0.047   86.4   0.0 0.0 ; semimajor axis, eccentricity, inclination, node, ap of
0.0  ; obj     number of mission related debris per launch
; (GLOBALSTAR)
1998 ; iear    year of first launch
20   ; life    planned lifetime of the constellation
56   ; nsat    number of satellites in the constellation
4    ; neach   number of satellites for each launch
4    ; la1     launches/year in the period of construction of the constellation
1    ; la2     launches/year to maintain the constellation
0    ; areaco  area (in m^2) of each satellite
450  ; dmasco  mass (in kg) of each satellite
7792  0.00   52.0   0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap
1    ; ndeo    deorbiting of old satellites? (1=yes, 0=no)
0    ; deo_pol deorbiting policy (0 = standard;  1 = non standard)
8400  0.00   52.0  0.0  ; non-standard deorbiting orbit (if deo_pol=1)
10.  0.001  0.1  0.1 ; spread around the nominal storage orbit
0.2  ; fail    failure rate [number of satellites/year]
1998 ;  upper stage left in orbit and year of change policy
20.   ; areuco  area (in m^2) of the upper stage for the constellation
1000. ; dmauco  mass (in kg) of the upper stage for the constellation
```

```
7151   0.090  52.0   0.0  0.0  ; semimajor axis, eccentricity, inclination, node, ap of
0.0  ; obj     number of mission related debris per launch
; (ORBCOM)
1997 ; iear    year of first launch
20   ; life    planned lifetime of the constellation
48   ; nsat    number of satellites in the constellation
8    ; neach   number of satellites for each launch
1    ; la1     launches/year in the period of construction of the constellation
1    ; la2     launches/year to maintain the constellation
0    ; areaco  area (in m^2) of each satellite
43   ; dmasco  mass (in kg) of each satellite
7203  0.00   45.    0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap
1    ; ndeo    deorbiting of old satellites? (1=yes, 0=no)
0    ; deo_pol deorbiting policy (0 = standard;  1 = non standard)
6758   0.00   86.4  0.0  ; non-standard deorbiting orbit (if deo_pol=1)
10.  0.001  0.1  0.1 ; spread around the nominal storage orbit
0.2  ; fail    failure rate [number of satellites/year]
1997 ;  upper stage left in orbit and year of change policy
0     ; areuco  area (in m^2) of the upper stage for the constellation
17.   ; dmauco  mass (in kg) of the upper stage for the constellation
6843.   0.049   45.  0.0  0.0  ; semimajor axis, eccentricity, inclination, node, ap of
0.0  ; obj     number of mission related debris per launch
; (GPS/NAVSTAR)
1978 ; iear    year of first launch
250  ; life    planned lifetime of the constellation
24   ; nsat    number of satellites in the constellation
1    ; neach   number of satellites for each launch
4    ; la1     launches/year in the period of construction of the constellation
1    ; la2     launches/year to maintain the constellation
0    ; areaco  area (in m^2) of each satellite
990. ; dmasco  mass (in kg) of each satellite
26528.  0.00   53.    0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap
1    ; ndeo    deorbiting of old satellites? (1=yes, 0=no)
0    ; deo_pol deorbiting policy (0 = standard;  1 = non standard)
27028   0.00   53.    0.0  ; non-standard deorbiting orbit (if deo_pol=1)
10.  0.001  0.1  0.1 ; spread around the nominal storage orbit
0.2  ; fail    failure rate [number of satellites/year]
1978 ;  upper stage left in orbit and year of change policy
0    ; areuco  area (in m^2) of the upper stage for the constellation
250. ; dmauco  mass (in kg) of the upper stage for the constellation
16240.   0.593   39    0.0  0.0  ; semimajor axis, eccentricity, inclination, node, ap o
0.0  ; obj     number of mission related debris per launch
```

```
; (GLONASS)
1982 ; iear    year of first launch
40   ; life    planned lifetime of the constellation
24   ; nsat    number of satellites in the constellation
3    ; neach   number of satellites for each launch
2    ; la1     launches/year in the period of construction of the constellation
1    ; la2     launches/year to maintain the constellation
0    ; areaco  area (in m^2) of each satellite
1480. ; dmasco  mass (in kg) of each satellite
25508.  0.0   64.8    0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap
1    ; ndeo    deorbiting of old satellites? (1=yes, 0=no)
0    ; deo_pol deorbiting policy (0 = standard;  1 = non standard)
25508   0.0   64.8  0.0  ; non-standard deorbiting orbit (if deo_pol=1)
10.  0.001  0.1  0.1 ; spread around the nominal storage orbit
0.4  ; fail    failure rate [number of satellites/year]
1982 ;  upper stage left in orbit and year of change policy
0.   ; areuco  area (in m^2) of the upper stage for the constellation
2500. ; dmauco  mass (in kg) of the upper stage for the constellation
25498.   0.0  64.8    0.0  0.0  ; semimajor axis, eccentricity, inclination, node, ap of
0.0  ; obj     number of mission related debris per launch
;  (GALILEO)
2005 ; iear    year of first launch
220  ; life    planned lifetime of the constellation
30   ; nsat    number of satellites in the constellation
1    ; neach   number of satellites for each launch
6    ; la1     launches/year in the period of construction of the constellation
1    ; la2     launches/year to maintain the constellation
0    ; areaco  area (in m^2) of each satellite
1000 ; dmasco  mass (in kg) of each satellite
29994. 0.0  56.0  0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap
1    ; ndeo    deorbiting of old satellites? (1=yes, 0=no)
0    ; deo_pol deorbiting policy (0 = standard;  1 = non standard)
30994  0.0  56.0  0.0   ; non-standard deorbiting orbit (if deo_pol=1)
10.  0.001  0.1  0.1 ; spread around the nominal storage orbit
0.2  ; fail    failure rate [number of satellites/year]
2005 ;  upper stage left in orbit and year of change policy
0.   ; areuco  area (in m^2) of the upper stage for the constellation
1000. ; dmauco  mass (in kg) of the upper stage for the constellation
18436. 0.629 56.0  0.0  0.0 ; semimajor axis, eccentricity, inclination, node, ap of the
0.0  ; obj     number of mission related debris per launch
STRUCTURE
1    ; nst     # of large structures under construction
```

```
1998 ; iear    starting year of the contruction
25   ; nplast  planned operative lifetime of the structure (years)
6    ; last    # of servicing missions/year
1    ; objst   # of objects/mission released (i.e. not connected to the struct.)
6778  0.00   52.0    0.0   ; orbital parameters of the struct.
```

C.3 LCLASS.DAT

The new format of *LCLASS.DAT* contains the following columns:

1. semimajor axis of the PayLoad (P/L) [km]
2. eccentricity of P/L
3. inclination of P/L [degrees]
4. argument of perigee of P/L [degrees]
5. cross sectional area of P/L [m^2]
6. mass of P/L [kg]
7. yearly growth of the P/L mass (e.g. 0.01 means 1 % growth in the masses)
8. percentage of the total routine launches due to the class
9. number of payload per launch in that class
10. operational lifetime of the payload [years]
11. flag for the upper stages:
    0 no upper stages left in orbit
    1 one upper stage left in orbit (orbital elements following)
    2 two upper stages left in orbit (orbital elements following)
12. area of first upper stage (U/S) [m^2]
13. mass of first U/S [kg]
14. mass of solid propellant used by the first U/S [kg]
15. flag to indicate if the solid rocket motor is burn at perigee [0] or apogee [1]
16. time of implementation of mitigation measure for the first U/S (de-orbiting)
17. semimajor axis of first U/S [km]

18. eccentricity of first U/S
19. inclination of first U/S [degrees]
20. argument of perigee of the first U/S [degrees]
21. area of second upper stage (U/S) [m^2]
22. mass of second U/S [kg]
23. mass of solid propellant used by the first U/S [kg]
24. flag to indicate if the solid rocket motor is burn at perigee [0] or apogee [1]
25. time of implementation of mitigation measure for the second U/S (de-orbiting)
26. semimajor axis of second U/S [km]
27. eccentricity of second U/S
28. inclination of second U/S [degrees]
29. argument of perigee of the second U/S [degrees]
30. number of mission related objects per launch
31. time of implementation of mitigation measure for the mission related objects

The elements for each upper stage are mandatory, even if they are the same as the payloads. If an area is set equal to 0.d0, this means that there is no reliable information available and, therefore, the program calculates it from the selected area vs. mass relation.
In column 9, for the GEO satellites, if the lifetime is negative, this means that the satellite is not re-orbited after the end of the operational lifetime, but the value indicates that it is controlled throughout the lifetime; i.e., it is not included in the projectiles population in the collision probability calculations. After the end of the operational lifetime, the satellite is left in its orbit and its identity changed so that it comes back into the projectiles density.
If the mass of solid propellant is zero then no SRM slag is produced. I.e., no specific flag is used to simulate the production of the SRM slag. The event is triggered only by the value of the mass in this file.

Iclass_1.dat

C.4 SDM.DAT

This is main input file of SDM. It contains the control flags for the models used within the code and for the mitigation measures.

```
20      ; nyea   = time span of the simulation (years)
2       ; npr    = print step in years (i.e. a printout every npr step)
5       ; ia_m   = area vs mass relation
i       ; c3     = velocity model (h=high, l=low, i=intermediate)
c       ; c6     = mass distribution after collisions
                 (c = CNUCE, e = EVOLVE, q = EVOLVE 4)
0       ; ev_exp = if eq. 1 then use EVOLVE 4 model for explosions
0       ; hanada = if eq. 1 then use Handa's model for low velocity
                   collisions
c       ; nmom   = mass distribution for high-int. expl.
                   ( c = CNUCE, p=Parrin.)
1.0     ; pico   = collision coupling factor
0       ; iror   = generation of RORSAT-like events  (1 = yes ; 0 = no)
0       ; ipopad = additional initial population (1 = yes ; 0 = no)
0       ; iprop  = orbital propagator:
                   (0 = only DCP  1 = only FOP  2= FOP or DCP)
3       ; iout   = output flag:
                   (1= mass, 2 = diameter, 3 = mass and diameter)
%%%%%  DE-orbiting - RE-orbiting OPTIONS
2010    ; year of beginning of mitigation measures
e       ; deflag = de-orbiting on circular (c) or elliptical (e) orbit
25.     ; reslif = residual lifetime of the orbit after
                   de-orbiting [years]
f       ; au_fi  = fixed (f) or automatic (a) de-orbiting
                   altitude choice
1400.   ; perdeo = perigee altitude of the spacecraft which
                   are re-orbited in a LEO graveyard orbit [km]
2000.   ; deoalt = altitude of the LEO circular graveyard orbit [km]
20. 80. ; Lower and upper limit of the triangular distribution for the
          storage zones amplitude
```

C.5 SNAPSYEAR.DAT

In this input file the user can indicate for which years the printout of the whole running population is desidered.

```
# Years in which we want the snapshots of the running population
# If the value is 55555 we want ALL the years (warning: disk space !!)
```

```
# The last year is always written (unless the file is empty, in which
# case NO snapshot is required)
2011
2021
2031
2041
2051
2061
2071
2081
2091
2101
```

C.6 DENSINP.DAT

In SDM (Version 3.0) it is possible to use population input files with every format and number of columns (provided that the data required by the program are present or, alternatively, a default is assumed).
The information about the structure of the input data is contained in the file *densinp.dat*, shown below:

```
; The first line contains the input format string for CREDEN.F
; DEFAULT UNITS: a in [km], mass in [kg], diameter in [m],
;                area in [m^2], angles in [deg]
*
53  ; number of header lines
11  ; number of columns
1 ; position of the column with the identification flag
2 ; position of the column with the sampling factor
3 ; position of the column with the mass
205 ; position of the column with the area (area or area/mass or mass/area)
4 ; position of the column with the diameter
6 ; position of the column with the semimajor axis
7 ; position of the column with the eccentricity
8 ; position of the column with the inclination
9 ; position of the column with the argument of node
10 ; position of the column with the argument of perigee
11 ; position of the column with the mean anomaly
```

In the first two comment lines the default units of measure for the quantity listed in the population file are given. If any quantity in the population file is not in the default units it must by changed *inside the population file, before running the preprocessing routines.*

First, the file contains one line with the format string, according to which the data are written in the input file. Since all the columns in the input file are read, for sake of generality, as floating point numbers, and then eventually converted into integers, in the format string all the *In* must be translated into *Fn.0*. If the data in the input file are written in free format, the corresponding line in *densinp.dat* will contain only a star (*).
Following the format line there is one line with the number of the header lines in the input file and one line with the total number of columns, n_c, in the input file. Then there are n_c lines with the positions, in the input file, of the columns corresponding to the different quantities that can be treated by the density generation software. A typical population file can contain; an integer labeling the object, the sampling factor, the semimajor axis, the eccentricity, the inclination, the mass, the area, the diameter and, possibly, the three angular arguments (argument of the node, argument of the perigee and mean anomaly).
The only non self-explaining row is the one labeled as: "position of the column with the area (area or area/mass or mass/area)". Since in some initial population file the area may not be given directly, but, for example, in terms of area over mass ratio, the user has to specify which quantity is to be read, and in particular:

- if the number of the column, *indac*, is comprised between 0 and 99 then the quantity contained in the column *indac* is the area;
- if the number of the column, *indac*, is comprised between 100 and 199 then the quantity contained in the column *indac-100* is the area over mass;
- if the number of the column, *indac*, is comprised between 200 and 299 then the quantity contained in the column *indac-200* is the mass over area.

It is important to notice that only mass, area, diameter, semimajor axis, eccentricity and inclination are strictly necessary to run the pre-processing software. If one of the other quantities is missing in the population file then a 0 must be put as the position of the corresponding column. In the case of identification flag a progressive number will be assigned by the program. For the sampling factor the unity will be assumed. In the case the diameter is missing, then this information will not be used in the calculation of the densities, while the missing angular arguments (Ω, ω, M) will be generated randomly by the code.

C.7 CREDEN.INP

The file *creden.inp* contains some of the input data needed by the preprocessing software. In particular it looks like:

```
100     ; time span of the propagation
1       ; nfreq, write a file with the whole population elements
                 every nfreq years
2       ; prop, 0 = DCP only ,
                1 = FOP only ,
```

```
               2 = DCP + FOP according top the orbit
123    ; idum, seed of the random number generator
```

The flag *nfreq* specifies the frequency (in years) of the output containing the whole population, in terms of orbital elements. With the flag *prop*, the user can choose the orbital propagator to propagate the debris orbit by choosing:

- prop = 0 use only DCP
- prop = 1 use only FOP
- prop = 2 use either DCP or FOP according to the type of orbit

If the flag **prop** equals 2 then the code checks on the orbital elements of the object. If the object is in a circular Low Earth Orbit then DCP is used. Otherwise FOP is used. The limit on the semimajor axis and eccentricity discriminating between the two regimes are set by the user in the header file *prop_lim.h* (A.16)
FOP requires a dedicated input file, *input.fop* (C.8) to specify a number of parameters of the dynamical model used to integrate the orbits.
The random number generator seed is used in case the angular elements are not present and should be generated by **creden.f** (A.3) (see the explanation in Sec. C.6).

C.8 INPUT.FOP

The dynamical model used in the orbit propagation by FOP is specified in the following file:

```
    4                          L
    4                          M
    1                          IDENS
    0                          ITEXO
    1.D-4                      RELERR
    1.D-4                      ABSERR
    30.D0                      STEP (DAYS)
   1000.D0                     TEXO
 3.9860130D5                   GE
6378.144D0                     RE
.4178074216D-2                 RATE
 .8181922D-1                   ELLIP
  6468.144D0                   RATM
  3.11D-3                      RDENS  ** J77 TE=1000 K **
   400.D0                      RHT
  56.00D0                      SHT    ** J77 TE=1000 K **
 5.412D-3                      CSRP   (4.51D-3 IF CR=1)
    2.2D0                      CDRAG
```

```
.1327150000D12                      GS
.4902784714D4                       GM
   2    0    -0.1082627000000000E-02    0.0000000000000000E+00
   3    0     0.2536414000000000E-05    0.0000000000000000E+00
   4    0     0.1623349700000000E-05    0.0000000000000000E+00
   5    0     0.2260856700000000E-06    0.0000000000000000E+00
   2    1     0.1253181183594739E-08   -0.9089259599006975E+00
   3    1     0.2210754552070675E-05    0.7113710190004468E+01
   4    1     0.6762567429674137E-06   -0.1384115360111180E+03
   5    1     0.9450092585732550E-07   -0.1195499653096793E+03
   2    2     0.1810826519633979E-05   -0.1493506184688393E+02
   3    2     0.3718680361116607E-06   -0.1739846296814032E+02
   4    2     0.1692065669068947E-06    0.3113723721703223E+02
   5    2     0.1179526968065780E-06   -0.1324601576474761E+02
   3    3     0.2214036571503680E-06    0.2110296993627385E+02
   4    3     0.6045863268736177E-07   -0.3896398715215931E+01
   5    3     0.1683737761901600E-07   -0.5163521111611788E+02
   4    4     0.7666183085687986E-08    0.3055290293387146E+02
   5    4     0.2375325635202597E-08    0.4281779712915544E+02
   5    5     0.1682116820559145E-08   -0.1533917833955905E+02
```

L and M	indicate the maximum degree and order of the Earth gravity field expansion
IDENS	is a flag to set air density model: 0 an exponential density model is used 1 a modified Jacchia-Roberts model is used.
ITEXO	is a flag to set exospheric temperature when the modified Jacchia-Roberts model is used: 0 compute temperature using time varying simulated solar flux 1 constant temperature provided by the user.
RELERR	is the relative accuracy of the integrator (recommended values between 1.d-6 to 1.d-8).
ABSERR	is the absolute accuracy of the integrator (recommended values between 1.d-6 to 1.d-8).
STEP	is the time step used to print the output in days.
TEXO	is the exospheric temperature in kelvin degrees needed when ITEXO = 1.
GE and RE	are the Earth's GM parameter (in km^3/sec^2) and the Earth's equatorial radius in km.
RATE	is the rotation rate of the Earth (in deg/sec).
ELLIP	is the ellipticity of the Earth's equator.
RATM	is the radius of the earth including atmospheric blockage (in km). it is used to compute shadow entry and exit in solar radiation pressure computation. The recommended value is RE + 90 km.
RDENS	is the reference density at reference height RHT to be used by the exponential atmosphere model when IDENS equals 0 (in kg/km^3).
RHT	is the reference height for the exponential density model (in km).
SHT	is the scale height of the exponential density model in (km).
CSRP	is the solar radiation pressure constant (in kg km^{-1} sec^{-2}). The value recommended by ESOC (subroutine coot) is $0.00451 \cdot k$, where $k < 1$ for translucent material, $k = 1$ for black body and $k = 2$ for perfectly reflective material.
CDRAG	is the drag coefficient (the recommended value is 2.2).
GS and GM	are the GM parameter of the Sun and the Moon, respectively (in km^3/sec^2).

The last lines of the file contains the values of the coefficients of the geopotential.

References

[1] Anselmo, L., Cordelli A., Pardini C. and Rossi A., 2000. *Final Report, Space Debris Mitigation: Extension of the SDM Tool* ESA/ESOC Contract No. 13037/98/D/IM, Consorzio Pisa Ricerche, Pisa, Italy.

[2] Johnson, N.L., Krisko P.H., Liou J.C. and Anz-Meador P.D., 2001. NASA's New Breakup Model of EVOLVE 4.0, *Adv. Space Res.*, **28**, No. 9, 1377-1384.

[3] Hanada, T., P. Krisko, P. Anz-Meador, N. Johnson, 2000. Consequences of continued growth in the GEO and GEO disposal orbital regimes, *paper AIAA-00-IAA.6.6.06*, 51^{st} IAF Congress, 2–6 Oct. 2000, Rio de Janeiro, Brasil.

[4] Anselmo, L., A. Cordelli, P. Farinella, C. Pardini and A. Rossi, 1996. Final Report, Study on Long Term Evolution of Earth Orbiting Debris, *Study Note of ESA/ESOC Contract No. 10034/92/D/IM(SC)*, Consorzio Pisa Ricerche, Pisa, Italy.

[5] Anselmo, A., A. Cordelli, C. Pardini and A. Rossi, 2000. Final Report: Space Debris Mitigation - Extension of the SDM tool, *Study Note of ESA/ESOC Contract No. 13037/98/D/IM*, Consorzio Pisa Ricerche, Pisa, Italy.

[6] *Meteoroid and Space Debris Terrestrial Environment Reference Model, MASTER-2001*, ESA SD-CD07, Release 1.0, December 2002.

[7] Rossi, A., Energetic cost and viability of the proposed space debris mitigation measures, 2002. *Journal of Spacecraft and Rockets*, **39**, No. 4, 540–550.

[8] Valsecchi, G.B., A. Rossi, and P. Farinella, 2000. Visualizing impact probabilities of space debris, *Space Debris*, **1**, 143–158. 4 2.3, 2.3.6 2.4, 2.4 1, 2.1, 2.3.6 1, 2.3.6 2.2 2.5 3.2, 3.2, 3.3, 3.3