Michael Koller

Markov Model

October 12, 2012

PUBLIC

Contents

Chapter 1
Application of the Markov model to Life Insurance

1.1 Traditional Rating of Life Contracts

Before starting with the Markov model, I would like to summarise how traditional calculations using commutation functions are performed. Usually one starts with the probabilities of death and then calculates a decrement table starting with, say, 100000 persons at age 20.

After that one, has to calculate the different commutation functions, which I assume everybody knows by heart. These numbers depend on the persons alive and on the technical interest rate i. Only when you have done this it is (in the classical framework) possible to calculate the necessary premiums. In the following we will look a little bit closer at the calculation of a single premium for an annuity. To do this we need the following commutation functions:

$$D_x = v \times l_x \text{where } l_x \text{ denotes the number of persons alive at age } x.$$
$$C_x = v \times (l_{x+1} - l_x)$$

Having this formalism it is well known that

$$\ddot{a}_x = \frac{N_x}{D_x}$$

From this example is easily seen that almost all premiums can be calculated by summation and multiplication of commutation functions. Such an approach has its advantages in an environment where calculations have to be performed by hand, or where computers are expensive. Calculation becomes messy if benefits are considered with guarantees or with refunds.

The Markov model here presented offers rating of life contracts without using commutation functions. It starts with calculation of the reserves and uses the involved probabilities directly. In order to see such a calculation let's review the above-mentioned example: We will use ${}_np_x$ to denote the probability of a person aged exactly x surviving for n years.

$$\begin{aligned}\ddot{a}_x &= \sum_{j=0}^{\infty} {}_jp_x \times v^j \\ &= 1 + p_x \times \ddot{a}_{x+1}\end{aligned}$$

The above formula gives us a recursion for the mathematical reserves of the contract. Hence one can calculate the necessary single premiums just by recursion. In order to do this, we need an initial condition, which is in our case $V_\omega = 0$.

The interpretation of the formula is easy: The necessary reserve at age x consists of two parts:

1. The annuity payment, and
2. The necessary reserve at age x+1. (These reserves must naturally be discounted.)

It should be pointed out that the calculation does not need any of the commutation functions; only p_x and the discount factor v are used. As a consequence this method does not produce the overheads of traditional methods.

In the following paragraphs the discrete time, discrete state Markov model is introduced and solutions of some concrete problems are offered.

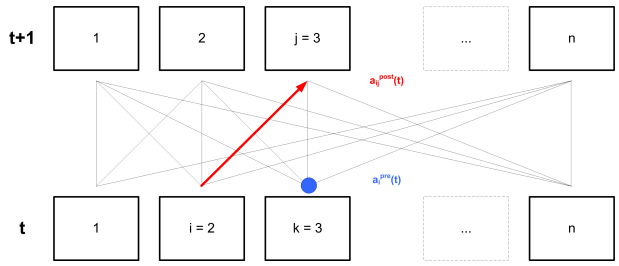
At this point, it is necessary to stress the fact that the following frame work can be used, with some modifications, in an environment with stochastic interest. But as we are limited in space and time we have to restrict ourselves to deterministic constant discount rates.

1.2 Life Insurance considered as Random Cash flows

The starting point of the Markov model is a set of states, which correspond to the different possible conditions of the insured persons. In life insurance the set of states usually consists of alive, dead. The set of states will be denoted by S.

The second point which originates from the life contract has to do with the so-called contractual functions which depend on the states and the time. Hence the structure of a generalised life contract can be thought of:

Contractual situation between time t and time $t+1$



From the above diagram it can be seen that a finite number of states is considered, and that for each transition $i \to j$ two different sums are paid, namely $a_{ij}^{\text{Post}}(t)$ at the end of the considered time interval and $a_i^{\text{Pre}}(t)$ at the beginning of it. It is clear that the value of the payment stream $a_{ij}^{\text{Post}}(t)$ has to be discounted by v in order to be compatible with $a_i^{\text{Pre}}(t)$. Probably it is worth remarking that the use of the two payment streams $a_i^{\text{Pre}}(t)$ and $a_{ij}^{\text{Post}}(t)$ eases the solution of things like payments during the year and the distinction between lump sums (generally payable at the end of the period) and annuities (at the beginning). Finally it must be said that premiums payable to the insurer can (not must (!)) be considered as benefits with the opposite sign.

Until now we have defined the sums which are payable if a certain insured event occurs. Now there has to be a probability law in order to rate the different transitions. In the following we denote by $p_{ij}(t, t+1)$ the probability of transition at time t from state $i \to j$. Hence in the language of the above diagram there is one transition probability assigned to each line between two states.

So summarising a Markov life insurance model consists of the following:

S	A finite state space (set).
$((p_{ij}(t))_{(i,j)\in S^2})_{t\in(1,2,\dots\omega)}$	The transition probabilities describing the Markov chain X_t on S.
$((a_i^{\text{Pre}}(t))_{i\in S})_{t\in(1,2,\dots\omega)}$	The prenumerando benefits relating, paying at the beginning of the corresponding period.
$((a_{ij}^{\text{Post}}(t))_{(i,j)\in S^2})_{t\in(1,2,\dots\omega)}$	The postnumerando benefits relating, paying at the end of the corresponding period, if a transition $i \to j$ happens.
$((v_i(t))_{i\in S})_{t\in(1,2,\dots\omega)}$	The yearly discount rate from $[t, t+1[$. We have $v_t = \sum_{j\in S} I_j(t)\, v_i(t)$.

1.3 Reserves, Recursion and Premiums

One of the most important quantities in actuarial science is the prospective reserve, as the insurer must have this amount of money for each policy. Therefore the concept of the prospective reserve is known to all actuaries. It is defined to be the present value of the future cash flow A given the information at present. Formally we write

$$V_j^+(t, A) := E[V(t, A \times \chi_{]t,\infty]}) \,|\, X_t = j],$$

(where j denotes the state at time t). This notation tells us, that the reserve depends heavily on the state of the policy.

In the context of the above we have

$$\begin{aligned}
\Delta A(t) &= \sum_{j\in S} I_j(t) \times a_i^{\text{Pre}}(t) + \sum_{(i,j)\in S\times S} \Delta N_{ij}(t) \times a_{ij}^{\text{Pre}}(t),\\
A(t) &= \sum_{k\le t} \Delta A(k),\\
\Delta V(t, A) &= v(t)\Delta A(t),\\
&= v(t)\left[\sum_{j\in S} I_j(t) \times a_i^{\text{Pre}}(t) + \sum_{(i,j)\in S\times S} \Delta N_{ij}(t) \times a_{ij}^{\text{Pre}}(t)\right],\\
v(t) &= \prod_{\tau\le t}\left[\sum_{j\in S} I_j(\tau) \times v_j(\tau)\right].
\end{aligned}$$

The direct calculation of the necessary reserves for the different states is not too easy if you consider a general time continuous Markov model. An advantage of this model is the existence of a powerful backwards recursion. The following formula (Thiele difference equation) allows the recursive calculation of the necessary reserves and hence of the necessary single premiums:

$$V_i^+(t) = a_i^{Pre}(t) + \sum_{j\in S} v_i(t)\, p_{ij}(t)\big\{a_{ij}^{Post}(t) + V_j^+(t+1)\big\}. \tag{1.1}$$

The interpretation of the formula is almost the same as in the trivial example at the beginning. In principle the present reserve consists of payments due to the different possible transitions and the discounted values of the future

necessary reserves. It can be seen that the above recursion uses only the different benefits, the probabilities and the discount factor. In order to calculate the reserve for a certain age one has to do a backwards recursion starting at the expiration date of the policy. For annuities this is usually the age ω when everybody has died. Starting the recursion it is necessary to have boundary conditions, which depend on the payment stream at the expiration date. Usually the boundary conditions are taken to be zero for all reserves. It should be pointed out that one has to do this recursion for the reserves of all states simultaneously.

After the calculation of the different reserves one can naturally determine the corresponding necessary single premiums by the principle of equivalence.

We want to end this section with a short proof of the above mentioned Thiele recursion:

We know that $A(t) = \sum_{k \leq t} \Delta A(k)$ and also that

$$\Delta V(t, A) = v(t) \left[\sum_{j \in S} I_j(t) \times a_i^{\text{Pre}}(t) + \sum_{(i,j) \in S \times S} \Delta N_{ij}(t) \times a_{ij}^{\text{Pre}}(t) \right].$$

Hence we have

$$\begin{aligned} V_i^+(t) &= \frac{1}{v(t)} \mathbb{E} \left[\sum_{\tau=t}^{\infty} v(\tau) \times \Delta A(\tau) \,|\, X_t = i \right] \\ &= \frac{1}{v(t)} \mathbb{E} \left[\sum_{j \in S} I_j(t+1) \times \sum_{\tau=t}^{\infty} v(\tau) \times \Delta A(\tau) \,|\, X_t = i \right], \end{aligned}$$

remarking that $\sum_{j \in S} I_j(t+1) = 1$. If we now consider all the terms in $\Delta A(t)$ for a given $I_j(t+1)$ for $j \in S$, it becomes obvious that the Markov chain changes from $i \to j$ and in consequence only $N_{ik}(t)$ increases by one for $k = j$. If we furthermore use the projection property and the linearity of the conditional expected value and the fact that $\mathbb{E}\left[I_j(t+1) \,|\, X_t = i\right] = p_{ij}(t, t+1)$, together with the Markov property, we get the formula if we split $V_i^+(t)$ as follows:

$$\begin{aligned} V_i^+(t) &= \frac{1}{v(t)} \mathbb{E} \left[\sum_{\tau=t}^{\infty} v(\tau) \times \Delta A(\tau) \,|\, X_t = i \right] \\ &= \frac{1}{v(t)} \mathbb{E} \left[\left\{ \sum_{\tau=t}^{t} + \sum_{\tau=t+1}^{\infty} \right\} v(\tau) \times \Delta A(\tau) \,|\, X_t = i \right]. \end{aligned}$$

Doing this decomposition we get for the first part:

$$\text{Part}_1 = a_i^{Pre}(t) + \sum_{j \in S} v_i(t)\, p_{ij}(t) a_{ij}^{Post}(t),$$

and for the second:

$$\text{Part}_2 = \sum_{j \in S} v_i(t)\, p_{ij}(t) V_j^+(t+1).$$

Adding the two parts together we get the desired result:

$$V_i^+(t) = a_i^{Pre}(t) + \sum_{j\in S} v_i(t)\, p_{ij}(t)\big\{a_{ij}^{Post}(t) + V_j^+(t+1)\big\}.$$

More concretely we have

$$\begin{aligned}
V_i^+(t) &= \frac{1}{v(t)}\mathbb{E}\left[\sum_{j\in S} I_j(t+1)\times\sum_{\tau=t}^{\infty} v(\tau)\times \Delta A(\tau)\,|\,X_t = i\right]\\
&= a_i^{Pre}(t) + \sum_{j\in S}\mathbb{E}\left[I_j(t+1)\times\sum_{\tau=t}^{\infty}\frac{v(\tau)}{v(t)}\times\Delta A(\tau)\,|\,X_t=i\right]\\
&= a_i^{Pre}(t) + \sum_{j\in S}\mathbb{E}\Bigg[I_j(t+1)v_i(t)\Bigg\{a_{ij}^{Post} +\\
&\quad +\mathbb{E}\Bigg[\sum_{\tau=t+1}^{\infty}\frac{v(\tau)}{v(t+1)}\times\Delta A(\tau)\,\Bigg|\,X_t=i, X_{t+1}=j\Bigg]\Bigg\}\,\Bigg|\,X_t=i\Bigg]\\
&= a_i^{Pre}(t) + \sum_{j\in S} v_i(t)\, p_{ij}(t)\big\{a_{ij}^{Post}(t) + V_j^+(t+1)\big\}.
\end{aligned}$$

We remark that this section can only be a short introduction to this topic and we refer to [**?**] for a more extensive discussion.

Chapter 2
Beispiele und Probleme aus der Praxis

2.1 Einleitung

In diesem Kapitel wollen wir einige Probleme aus der Praxis genauer untersuchen. Zudem dienen die Beispiele dazu, die Möglichkeiten des Markovmodells zu illustrieren und ein paar Tricks für die Modellierung aufzuzeigen. Angesichts der Tatsache, dass in der Praxis hauptsächlich das diskrete Modell verwendet wird, wollen wir die Beispiele auf diesem Modell aufbauen.

Neben den Gegebenheiten, welche durch das Modell induziert werden, müssen auch die Usanzen, welche aus der Praxis resultieren, beachtet werden. Dies hängt einerseits damit zusammen, dass man ein Modell anstrebt, welches auch die Vergangenheit abbilden kann. Andererseits ist es oft so, dass bestimmte Formeln aufgrund von Abmachungen fixiert sind. So kommt es nicht von ungefähr, dass wir in diesem Kapitel auch gegebene Formeln mittels der Vertragsfunktionen nachvollziehen wollen.

2.2 Unterjährige Zahlungen

Als Erstes wollen wir uns dem Problem der unterjährigen Renten zuwenden. Um dieses Problem zu verstehen, muss man wissen, dass die in der Praxis betrachtete Zeitdifferenz normalerweise 1 Jahr beträgt. Andererseits werden Altersrenten oft unterjährig ausbezahlt. Wir wollen im Folgenden eine $4/4$ vorschüssige Altersrente betrachten. Dies bedeutet, dass der Versicherungsnehmer alle 3 Monate eine Rente der Höhe $\frac{1}{4}$ erhält. Wir nehmen für den Moment an, dass die Sterbewahrscheinlichkeit für dieses Jahr q_x beträgt, und dass die Sterbewahrscheinlichkeit $q_x^{[4]}$ für das Vierteljahr dem folgenden Gesetz folgt:

$$(1 - q_x^{[4]})^4 = 1 - q_x.$$

Dies bedeutet, dass die Sterbeintensität während des ganzen Jahres konstant ist. Im zeitdiskreten Modell, bei welchem wir einen Zeitschritt von 3 Monaten voraussetzen, ist die laufende, vorschüssige Rente durch die folgende Vertragsfunktion gegeben:

$$a_*(t) = \frac{1}{4}.$$

Andererseits gilt die Rekursion:

$$V_*(t) = a_*(t) + (1 - q_x^{[4]})v^{\frac{1}{4}}\, V_*(t + \frac{1}{4}).$$

Wir leiten jetzt die Rekursion für ein ganzes Jahr her. Es gelten die folgenden Gleichungen:

$$V_*(t) = a_*(t) + (1 - q_x^{[4]})v^{\frac{1}{4}}\, V_*(t + \frac{1}{4})$$

$$
\begin{aligned}
&= a_*(t) \times \sum_{k=0}^{3} \left((1 - q_x^{[4]}) v^{\frac{1}{4}} \right)^k + (1 - q_x) v \, V_*(t+1) \\
&= a_*(t) \times \frac{1 - (1 - q_x)v}{1 - \left((1 - q_x^{[4]}) v^{\frac{1}{4}} \right)} + (1 - q_x) v \, V_*(t+1) \\
&\approx \frac{5}{8} + (1 - q_x) v \left(\frac{3}{8} + V_*(t+1) \right),
\end{aligned}
$$

wobei wir bei dem letzten Schritt die Taylorentwicklung von f um $z = 1$ benutzt haben:

$$
\begin{aligned}
f(z) &= \frac{1}{4} \left(1 + z^{0.25} + z^{0.50} + z^{0.75} \right), \\
f(1) &= 1, \\
\frac{d}{dz} f(z)|_{z=1} &= \frac{3}{8}, \\
f(z) &\approx 1 + \frac{3}{8} (z - 1) \\
&= \frac{5}{8} + \frac{3}{8} z.
\end{aligned}
$$

Aus dem obigen Beispiel haben wir gesehen, wie unterjährige Zahlungen in einem Modell mit Zeitschrittweite 1 Jahr behandelt werden können. Es sei an dieser Stelle angemerkt, dass die oben hergeleitete Approximation genau derjenigen entspricht, welche normalerweise in einem Modell mit Kommutationszahlen benutzt wird. Weiterhin soll angemerkt werden, dass sich die obigen Überlegungen vollständig auf Versicherungen auf zwei Leben usw. übertragen lassen.

Exercise 1 *1. Wie kann das obige Verfahren bei einem Zeitintervall von 1 Jahr auf eine anwartschaftliche Invalidenrente mit 3 Monaten Wartefrist übertragen werden?*

2. *Berechnen Sie die entsprechenden Approximationen für eine sofort beginnende, vierteljährlich vorschüssige Altersrente auf zwei Leben (für den Zustand $**$).*
3. *Das obige Beispiel kann auch gelöst werden, indem man versucht, die exakte Lösung in zwei Terme der Form $(1 - q_x)$ bzw. v zu entwickeln. Wie lautet die Lösung in diesem Fall?*

Beachten Sie, dass es bei dem obigen Vorgehen nicht unbedingt erforderlich ist, dass alle unterjährigen Zahlungen dieselbe Höhe haben.

Example 2 *Im folgenden Beispiel berechnen wir den Fehler des Deckungskapitals, welcher durch die Approximation für eine vierteljährlich vorschüssige Rente entsteht. Die Sterbewahrscheinlichkeiten entsprechen (**??**). Die Resultate der Berechnung finden sich in Tabelle 2.1. Hieraus ist ersichtlich, dass der Fehler für den normalen Altersbereich unter 85 Jahren sehr klein bleibt.*

2.3 Garantierte Renten

Als Nächstes wollen wir uns kurz dem Problem der garantierten Renten zuwenden. Dieser Typ der Altersrente erfüllt das Bedürfnis, bei einem vorzeitigen Tod nicht alles zu verlieren. Dies bedeutet, dass der Versicherungsnehmer mit dem Eintritt in den Rentenbezug die garantierte Anwartschaft auf eine bestimmte Anzahl von Renten erhält. Technisch entspricht dieses Versprechen der Anpassung der Sterbewahrscheinlichkeiten für die versicherte Person während der Garantiezeit.

Dieses Problem kann jedoch auch wie folgt gelöst werden. Wir gehen hierzu von einer garantierten Altersrente (Garantiedauer ab 65 für 10 Jahre) aus, welche ab dem 65. Lebensjahr gezahlt wird. Für die normale Altersrente

Table 2.1 Fehler bei Renten durch Approximation der unterjährigen Zahlungen

x	Anteil p.a. exakt	Anteil p.a. approx.	Total exakt	Total approx.	Fehler Total
114	0.4436	0.6421	0.4436	0.6421	44.7533%
113	0.5298	0.6681	0.5808	0.7420	27.7533%
112	0.5874	0.6922	0.6915	0.8253	19.3363%
111	0.6323	0.7145	0.7973	0.9115	14.3198%
110	0.6692	0.7351	0.9033	1.0027	11.0058%
105	0.7917	0.8170	1.4893	1.5472	3.8884%
100	0.8614	0.8724	2.2214	2.2597	1.7228%
95	0.9047	0.9098	3.1358	3.1630	0.8654%
90	0.9325	0.9351	4.2484	4.2687	0.4773%
85	0.9508	0.9521	5.5575	5.5733	0.2846%
80	0.9629	0.9636	7.0436	7.0564	0.1817%
75	0.9709	0.9714	8.6713	8.6820	0.1231%
70	0.9763	0.9766	10.3937	10.4028	0.0879%
65	0.9799	0.9801	12.1588	12.1667	0.0656%
60	0.9823	0.9825	13.9156	13.9227	0.0508%
50	0.9850	0.9851	17.2341	17.2399	0.0335%

sind die nichttrivialen Vertragsfunktionen gegeben durch

$$a_*(t) = \begin{cases} 0, \text{ falls } t < 65, \\ 1, \text{ falls } t \geq 65. \end{cases}$$

Betrachtet man nun eine garantierte Altersrente, ist es nötig, den Zustand † zu unterteilen in Tod vor 65 (symbolisch: $\dagger_<$) und in Tod nach 65 ($\dagger_\geq$). In diesem Fall lauten die massgebenden Übergangswahrscheinlichkeiten wie folgt:

$$\begin{aligned} p_{**}(x) &= 1 - q_x, \\ p_{*\dagger_<}(x) &= \begin{cases} q_x, \text{ falls } t < 65, \\ 0, \text{ falls } t \geq 65, \end{cases} \\ p_{*\dagger_\geq}(x) &= \begin{cases} 0, \text{ falls } t < 65, \\ q_x, \text{ falls } t \geq 65, \end{cases} \\ p_{\dagger_<\dagger_<}(x) &= 1, \\ p_{\dagger_\geq\dagger_\geq}(x) &= 1. \end{aligned}$$

Die nichttrivialen Vertragsfunktionen lauten nun wie folgt: (wir gehen von einer $1/1$ vorschüssigen Altersrente aus.)

$$\begin{aligned} a_*(t) &= \begin{cases} 0, \text{ falls } t < 65, \\ 1, \text{ falls } t \geq 65, \end{cases} \\ a_{\dagger_\geq}(t) &= \begin{cases} 1, \text{ falls } t \in [65; 75[, \\ 0, \text{ sonst.} \end{cases} \end{aligned}$$

Zur Illustration betrachten wir zwei Beispiele für garantierte Renten.

Example 3 *Das folgende Beispiel soll den Verlauf des Deckungskapitals für einen 65jährigen Mann illustrieren. Wir gehen von einer 15jährigen Garantiezeit aus: (Sterblichkeit nach GRM 1995,* $1/1$*-vorschüssig,* $\omega = 121$*)*

Alter	**Einlagesatz mit Garantie**	**Einlagesatz ohne Garantie**	**Verhältnis in %**
121	*10000*	*10000*	*100 %*
120	*14694*	*14694*	*100 %*
110	*27143*	*27143*	*100 %*
100	*42031*	*42031*	*100 %*
90	*65298*	*65298*	*100 %*
80	*91840*	*91840*	*100 %*
75	*124057*	*107387*	*116 %*
70	*151184*	*124817*	*121 %*
65	*174024*	*142454*	*122 %*

BILD: AGBILD1 *Abbildung* **??** *zeigt das Deckungskapital einer temporären (20 Jahre) für 10 Jahre garantierten, sofort beginnenden Altersrente.*

2.4 Rückgewähr

Die Erlebensfallversicherungen mit Rückgewähr stellen eine besondere Art der Versicherung dar, bei welcher der Versicherungsnehmer bei dem Todesfall einen Teil der einbezahlten Prämien zurückerhält. Die Arten der Rückgewähr umfassen unter anderem folgende Typen:

1. Rückgewähr der bezahlten Prämien vor Fälligkeit der Altersrente,
2. Rückgewähr der bezahlten Prämien abzüglich der ausbezahlten Leistungen, (Vollständige Rückgewähr)
3. Rückgewähr des vorhandenen Deckungskapitals vor oder auch während der Fälligkeit der Altersrente.

Die ersten beiden Arten der Rückgewähr kann man sich als zusätzliche Todesfalldeckung vorstellen. Diese Typen der Rückgewähr werden von der Versicherungsindustrie schon seit langem verkauft. Dies ist auch der Grund, weshalb wir uns auf die dritte Art der Rückgewähr konzentrieren wollen. Auch für diese Art der Rückgewähr sind verschiedene Ausgestaltungen denkbar.

Example 4 (Rückgewähr des Deckungskapitals) *Bevor wir mit der Tarifierung dieser Versicherung beginnen, stellen wir uns die Situation vor, bei welcher im Todesfall das Deckungskapital als Rückgewährsumme versichert ist. In diesem Fall gilt die folgende Rekursion:*

$$V_*(x) = 1 + p_{**}(x)\ v\ V_*(x+1) + p_{*\dagger}(x)\ V_*(x).$$

(Wir nehmen hier an, dass das Deckungskapital für die Rückgewähr zu Beginn der Periode ausbezahlt werde.) Wir erhalten durch eine einfache Umformung die folgende modifizierte Rekursion:

$$V_*(x) = \frac{1 + p_{**}(x)\, v\, V_*(x+1)}{(1 - p_{*\dagger}(x))}.$$

Mit der obigen Formel haben wir das Problem für den Fall der Einmaleinlage gelöst. Für den Fall von periodischen Prämien ist es nötig, eine kompliziertere Gleichung zu lösen. Dies geschieht am einfachsten mit Hilfe numerischer Methoden.

Tabelle 2.2 zeigt einen Vergleich der Einmaleinlagen für die verschiedenen Möglichkeiten der Rückgewähr bei Altersrenten. Bei der Rückgewähr des Deckungskapitals endet diese mit der Fälligkeit der Rente. Abbildung 2.2 zeigt denselben Vergleich im Fall von prämienpflichtigen Versicherungen.

BILD: ATWBILD1

Table 2.2 Vergleich verschiedener Arten der Rückgewähr (KT 1995, $s = 65$, Mann)

Alter	Rückgewähr des DK	Rückgewähr der Einlage	Vollständige Rückgewähr
40	5.86921	5.50680	5.58673
45	6.97078	6.64053	6.78605
50	8.27910	8.01191	8.27932
55	9.83297	9.65914	10.15667
60	11.67848	11.61117	12.55151
65	13.87038	13.87038	15.69276

Example 5 *Das nächste Beispiel ist etwas ausgefallener. Versichert ist eine anwartschaftliche Witwenrente gegen Einmaleinlage mit Rückgewähr des Deckungskapitals bevor der Mann ein Alter von 85 Jahren erreicht hat, bei dem Tod der Frau oder dem gleichzeitigen (innerhalb eines Jahres) Tod des Mannes und der Frau. In diesem Fall lautet die Rekursion wie folgt:*

$$V_{(**)}(x) = \frac{v\left(p_{(**)(**)}\, V_{(**)}(x+1) + p_{(**)(\dagger *)}\, V_{(\dagger *)}(x+1)\right)}{\left(1 - p_{(**)(*\dagger)}(x) - p_{(**)(\dagger\dagger)}(x)\right)}.$$

Abbildung **??** *zeigt die Lösung der obigen Gleichungen in grafischer Form. Hierbei wird deutlich sichtbar, dass die Rückgewähr nur bis zum Alter von 85 Jahren gewährt wird. Danach wird die normale Rekursionsgleichung verwendet.*

BILD: WTWBILD1

2.5 Kapitalversicherungen mit stochastischem Zins

In diesem Abschnitt wollen wir Versicherungen mit stochastischen Zinsen betrachten. Es soll hier darum gehen, die Methoden, welche wir angetroffen haben, ein wenig zu illustrieren. Bei den verwendeten Zinsmodellen geht es in erster Linie darum, Beispiele aufzuzeigen. Sie erheben keinen Anspruch darauf, die Realität widerzuspiegeln. Wir betrachten die Gemischte Versicherung aus Beispiel **??** und gehen von einem 30jährigen Mann aus, welcher eine Todesfallsumme der Höhe 200'000 Fr. versichert hat und im Erlebensfall 100'000 Fr. erhält. Wir wollen sowohl die Versicherung gegen Einmaleinlage als auch gegen Jahresprämie betrachten.

Es sollen die folgenden Zinsmodelle betrachtet werden:

1. Ein konstanter technischer Zins von 5%.
2. Ein Zinsmodell, welches einen zyklischen Wirtschaftsverlauf modelliert.
3. Ein Random-Walk-Modell für die Zinsen.

Example 6 (Zinsmodelle) *Um die obige Aufgabenstellung lösen zu können, müssen in einem ersten Schritt die verschiedenen Zinsmodelle ausgearbeitet werden. Zum konstanten Zinssatz ist nichts zu sagen.*

Zyklischer Wirtschaftsverlauf: *Wir gehen davon aus, dass es einen achtjährigen Wirtschaftszyklus gibt und modellieren das Zinsgeschehen wie folgt:*

Zustand	Bemerkung	Jahreszins	p_{ii}	p_{ii+1}	p_{ii+2}
0	*Ausgangslage*	*5.0 %*	*0.1*	*0.7*	*0.2*
1	*Steigender Zins*	*5.5 %*	*0.1*	*0.7*	*0.2*
2	*Max. Zins*	*6.0 %*	*0.1*	*0.7*	*0.2*
3	*Fallender Zins*	*5.5 %*	*0.1*	*0.7*	*0.2*
4	*Mittelwert*	*5.0 %*	*0.1*	*0.7*	*0.2*
5	*Fallender Zins*	*4.5 %*	*0.1*	*0.7*	*0.2*
6	*Min. Zins*	*4.0 %*	*0.1*	*0.7*	*0.2*
7	*Steigender Zins*	*4.5 %*	*0.1*	*0.7*	*0.2*

Dem obigen Modell zufolge bewegt sich der Zins in Zyklen, wobei der Zufall für einen beschleunigten oder verlangsamten Zyklus sorgt.

Random Walk: *Es wird ein Random-Walk-Modell betrachtet mit Modifikation an den Rändern:*

Zustand	Bemerkung	Jahreszins	p_{ii-1}	p_{ii}	p_{ii+1}
0	*Min. Zins*	*4.0 %*	*0.0*	*0.5*	*0.5*
1		*4.3 %*	*0.4*	*0.2*	*0.4*
2		*4.7 %*	*0.4*	*0.2*	*0.4*
3	*Ausgangslage*	*5.0 %*	*0.4*	*0.2*	*0.4*
4		*5.3 %*	*0.4*	*0.2*	*0.4*
5		*5.7 %*	*0.4*	*0.2*	*0.4*
6	*Max. Zins*	*6.0 %*	*0.5*	*0.5*	*0.0*

Wir gehen bei beiden Modellen davon aus, dass der aktuelle Zins 5% *beträgt.*

Als Nächstes wollen wir unser Modell dahingehend vereinfachen, dass wir den technischen Zinssatz nur für den Übergang $* \rightsquigarrow *$ stochastisch modellieren. Wir verwenden für den Übergang $* \rightsquigarrow \dagger$ also stets einen technischen Zinssatz von 5%.

Example 7 (Einlagen und Prämien) *Um die Einlagen und Prämien zu berechnen, ist es notwendig, die Rekursion für die verschiedenen Zinsmodelle durchzuführen:*

Konstanter Zins *In diesem Fall beträgt die Prämie* $P = 24755/16.77946 = 1475.30$ *Fr. p.a., und es ergeben sich die folgenden Resultate:*

Alter	Leistungs-barwert	Prämien-barwert	DK bei Einlage	DK bei Prämie
65	*100000*	*0.00000*	*100000*	*100000*
64	*96510*	*1.00000*	*96510*	*95035*
60	*83599*	*4.45585*	*83599*	*77026*
55	*69535*	*7.84428*	*69535*	*57963*
50	*57483*	*10.49434*	*57483*	*42000*
45	*47219*	*12.59982*	*47219*	*28631*
40	*38554*	*14.28589*	*38554*	*17478*
35	*31305*	*15.64032*	*31305*	*8231*
31	*25974*	*16.57022*	*25974*	*1528*
30	*24755*	*16.77946*	*24755*	*0*

Zyklischer Zins: *In diesem Fall beträgt die Prämie* $P = 24630/16.65234 = 1479.07$ *Fr. p.a.*

Alter	**Leistungs-barwert** $i_t = 4\%$	**Leistungs-barwert** $i_t = 5\%$	**Leistungs-barwert** $i_t = 6\%$	**DK bei Prämie** $i_t = 5\%$
65	*100000*	*100000*	*100000*	*100000*
64	*97414*	*95624*	*96510*	*95035*
60	*83854*	*83369*	*82536*	*76004*
55	*70204*	*68904*	*68923*	*57431*
50	*57834*	*57170*	*57128*	*41766*
45	*47482*	*46996*	*46812*	*28368*
40	*38832*	*38316*	*38233*	*17323*
35	*31517*	*31130*	*31072*	*8181*
31	*26145*	*25838*	*25761*	*1509*
30	*24914*	*24630*	*24558*	*0*

Random Walk: *In diesem Fall beträgt die Prämie* $P = 24936/16.81204 = 1483.20$ *Fr. p.a.*

Alter	**Leistungs-barwert** $i_t = 4\%$	**Leistungs-barwert** $i_t = 5\%$	**Leistungs-barwert** $i_t = 6\%$	**DK bei Prämie** $i_t = 5\%$
65	*100000*	*100000*	*100000*	*100000*
64	*97414*	*96510*	*95624*	*95027*
60	*86558*	*83611*	*80775*	*77002*
55	*73429*	*69588*	*65924*	*57951*
50	*61470*	*57581*	*53886*	*42008*
45	*50934*	*47356*	*43963*	*28652*
40	*41854*	*38717*	*35746*	*17503*
35	*34154*	*31482*	*28952*	*8247*
31	*28476*	*26155*	*23959*	*1531*
30	*27171*	*24936*	*22822*	*0*

2.6 Invaliditätsversicherungen

Wir wollen die Modellierung einer temporären Invaliditätsversicherung mit dem Markovmodell betrachten. Hier müssen zumindest die Zustände $\{*, \diamond, \dagger\}$ mit den entsprechenden Übergangswahrscheinlichkeiten betrachtet werden.

Da die Reaktivierungswahrscheinlichkeit massgebend von der abgelaufenen Dauer seit Invalidierung abhängt, ist es notwendig, den Zustand $\diamond$ weiter aufzuteilen in $\diamond_1, \diamond_2, \ldots, \diamond_n$, wobei wir mit $\diamond_k$ diejenigen Personen bezeichnen, welche zwischen $[k-1, k[$ Jahren invalid sind. Der Zustand $\diamond_n$ spielt eine besondere Rolle. Hier nehmen wir an, dass die Personen nicht mehr reaktivieren können. Die Problematik der unterschiedlichen Reaktivierungswahrscheinlichkeiten in Abhängigkeit zu der abgelaufenen Zeit wird durch Abbildung **??** verdeutlicht. Man kann beobachten, dass die Reaktivierungswahrscheinlichkeit kurz nach der Invalidierung noch hohe Werte annimmt, welche jedoch mit zunehmendem Alter, in welchem die Invalidität eintritt, zurückgehen.

BILD: Reakt

BILD: Invmodell

Auf der anderen Seite wird deutlich, dass die Reaktivierungswahrscheinlichkeit mit zunehmender Dauer seit der Invalidierung in etwa exponentiell abnimmt. Dieser starke Rückgang der Reaktivierungswahrscheinlichkeit, ausgehend von einem hohen Niveau, ist auch ein Grund, weshalb oft Wartefristen vereinbart werden. Diese führen bezüglich des Modells zu einer leichten Modifikation. Das Zustandsdiagramm für diesen Versicherungstyp wird in Abbildung **??** dargestellt.

Der Grund für die Aufteilung des Zustandes Invalidität ($\diamond$) in eine Menge von Zuständen $\diamond_1, \ldots, \diamond_n$ liegt in der Abhängigkeit der Reaktivierungswahrscheinlichkeit von der abgelaufenen Zeitdauer als Invalider. Hierbei wird angenommen, dass für den Zustand $\diamond_n$ keine Reaktivierung mehr stattfindet. Somit stellt sich für dieses Versicherungsmodell die Frage, wie gross n sein muss, damit der Fehler eine bestimmte Schranke unterschreitet. Um diese Grösse zu bestimmen, benutzen wir die folgenden Grundwahrscheinlichkeiten:

$$\begin{aligned}
p_{*\dagger}(x) &= \exp(-7.85785 + 0.01538x + 0.000577355x^2),\\
p_{*\diamond_1}(x) &= 3 \times 10^{-4} \times (8.4764 - 1.0985x + 0.055x^2),\\
p_{\diamond_k *}(x) &= \begin{cases} \exp(-0.94(k-1)) \times \alpha(x,k), & \text{falls} \quad k < n,\\ 0, & \text{sonst,} \end{cases}\\
\alpha(x,k) &= 0.773763 - 0.01045(x-k+1),\\
p_{\diamond_k \dagger}(x) &= 0.008 + p_{*\dagger}(x),\\
p_{**}(x) &= 1 - p_{*\diamond_1}(x) - p_{*\dagger}(x),\\
p_{\diamond_k \diamond_{k+1}}(x) &= 1 - p_{\diamond_k *}(x) - p_{\diamond_k \dagger}(x).
\end{aligned}$$

Für die Berechnungen gehen wir von einer $1/1$ vorschüssigen Invalidenrente mit Vertragsfunktionen

$$a^{\text{Pre}}_{\diamond_k}(x) = \begin{cases} 1, \ \text{falls} \ \ x < 65,\\ 0, \ \text{sonst,} \end{cases}$$

aus.

Das Deckungskapital für einen Aktiven und die verschiedenen n wird durch Abbildung **??** dargestellt.

BILD: Invmodell

Sucht man nun dasjenige n, für welches der Fehler für alle Altersstufen zwischen 25 und 65 kleiner als 5 % ist, ergibt sich etwa $n = 6$. Abbildung **??** zeigt die Schadenreserve für das Invaliditätsmodell mit $n = 6$ für verschiedene Alter.

BILD: Invmodell

Chapter 3

omarkov.h

```
//////////////////////////////////////////////////////////////////
//                                                              //
// Markovobjekt fuer LV Zahlungsstroeme                         //
//                                                              //
// Autor Michael Koller                                         //
//                                                              //
// Datum March 2011: erstellt                                   //
//                                                              //
//////////////////////////////////////////////////////////////////

#ifndef _OMARKOV_INCLUDED
#define _OMARKOV_INCLUDED

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include "tableserver.h"
#pragma message (" >>>>> More Default States")
/*
Die Werte sind gegeben wie folgt:
DEF_MAXTIMES  -- Standart max Zeit welche gebraucht wird
DEF_MAXSTATES -- Standart max Zustaende welche gebraucht wird
DEF_DEFNRMOMENTS -- Std wert fuer Anzahl Momente
DEF_STOC_INT     -- false = normales Zinsmodell, nur abhaengig vom
                    Ausgangszustand
Wenn deterministischer Zins, so steht der Zins für alle Elemente auf der Diagonale

MAXMOMENTE    -- Max Momente welche das Prg berechnen kann
*/
#define DEF_MAXTIMES     200l  // Das ist ein uebersteuerbarer Defaultwert
#define DEF_MAXSTATES     40l  // Das ist ein uebersteuerbarer Defaultwert
#define DEF_DEFNRMOMENTS   1l  // Das ist ein uebersteuerbarer Defaultwert
#define DEF_STOC_INT   false   // Das ist ein uebersteuerbarer Defaultwert
#define MAXMOMENTE        10l  // ! Absolute Schranke
#define WITH_RP
#undef  ALLOCATE_CF_TO_EXACT_TIMES  // Falls defiert werden CF zu den exakten Zeiten alloziert
#define MAX_TEX_ROWS      8
#define MAX_TEX_LINES     50
#define WITH_TeX
#define WITH_SIMUALATION      // Implementation not completed
#define TEX_TAB         "lrrrrrrrr"
#define TECHEPS 1.e-10
#define INCL_ANNMOD  // To include ANNMOD
```

```
// Die folgende Klasse implementiert einen verketteten Leistungs oder W'keits-vektor

class LV_VECTOR
{
public:
  // Bem - lVonIpt und INachIpt ist eine Info von wo nach wo (eg i to j)
  // Die Werte sind doubles
  LV_VECTOR(long lXDim, long lVonIpt, long lNachIpt);   // lXDim Laenge des Vect.
  ~LV_VECTOR();
  void            vReset();                             // Alles auf Null setzten
  double          dSetValue(long lX, double dValue);    // Wert setzten
  double          dAddValue(long lX, double dValue);    // Wert Addieren
  double          dGetValue(long lX);                   // Wert zurückerhalten
  void            vSetNextPtr(LV_VECTOR * psymNextIpt); // Pointer zum naechsten Element
  LV_VECTOR   * psymNext; // Das ist fuer die Verkettung, der Pointer zum naechsten
                          // Element, == NULL heisst letztes Element
  long            lVon;
  long            lNach;

private:
  long            lXMax;
  double      *   pdValues; // Das sind die eigentlichen Daten
};

class ILV_VECTOR
{
public:
  // Bem - lVonIpt und INachIpt ist eine Info von wo nach wo (eg i to j)
  // Die Werte sind long
  ILV_VECTOR(long lXDim, long lVonIpt, long lNachIpt);  // lXDim Laenge des Vect.
  ~ILV_VECTOR();
  void          vReset();                               // Alles auf Null setzten
  long          lSetValue(long lX, long lValue);        // Wert setzten
  long          lAddValue(long lX, long lValue);        // Wert Addieren
  long          lGetValue(long lX);                     // Wert zurückerhalten
  void            vSetNextPtr(LV_VECTOR * psymNextIpt); // Pointer zum naechsten Element
  LV_VECTOR   * psymNext; // Das ist fuer die Verkettung, der Pointer zum naechsten
                          // Element, == NULL heisst letztes Element
  long              lVon;
  long              lNach;

private:
  long          lXMax;
  long      *   plValues; // Das sind die Werte
};

class LV_MATRIX
{
public:
  LV_MATRIX(long lXDim, long lYDim, long lIdentIpt);          // lX/YDim Dimension Matrix
  ~LV_MATRIX();
  void            vReset();                                   // Werte auf Null setzten
  double          dSetValue(long lX, long lY, double dValue); // Wert belegen
  double          dAddValue(long lX, long lY, double dValue); // Addieren usw.
  double          dMultiplyValue(long lX, long lY, double dValue);
  double          dGetValue(long lX, long lY);
  void            vSetNextPtr(LV_MATRIX * psymNextIpt);
  LV_MATRIX   *   psymNext;
private:
  long            lXMax;
  long            lYMax;
  long            lIdent;
  double      *   pdValues;    // Pool aller Werte (doubles)
  double      **  ppdValues;   // Interne Matrix mit ppdValues[i][j] kann auf das entspr.
                               // Element zugegriffen werden
```

```
};

class MARKOVLV
{
public:
  /*  MARKOVLV(); */
  MARKOVLV(long lMaxTimesIpt, long lMaxStatesIpt, long lNrDefMomentsIpt); // Overrides Defaults
  ~MARKOVLV();
  void           vReset();   // Alles Zuruecksetzen
  void           vSetInternals(long lMaxTimes, long lMaxStates); // Diese Werte neu belegen
  void           vSetStartTime(long lTime); // Zeit an welcher Rekursion beginnt, zB 120
  void           vSetStopTime(long lTime);  // Zeit an welcher Rekursion stoppt zB 30
  void           vSetNrStates(long lNrStatesIpt); // Anzahl Zustaende des Modells
  void           vSetGetData(bool bStatus); // Falls true werden ueberschreiben die folgenden
  // 4 Funktionen keine Werte und geben die Werte nur zurueck
  // dSetPre - a_i^Pre(t)
  // dSetPost - a_{ij}^Post(t)
  // dSetPij  - p_{ij}(t)
  // dSetDisc - v_{i}(t) bzw v_{ij}(t) falls vSetInterestModel(true)
  double         dSetPre(long lTime, long lVon, long lNach, double dValue);  // lNach irrelevant
  double         dSetPost(long lTime, long lVon, long lNach, double dValue);
  double         dSetPij(long lTime, long lVon, long lNach, double dValue);
  double         dSetDisc(long lTime, long lVon, long lNach, double dValue);
  void           vSetInterestModel(bool bStocInterest);  // true heisst stochastischer Zins
  void           vSetDefaultNrMoments(long lNrMoments);  // Wenn man hoehere Momente will
  double         dGetDK(long lTime, long lState, long lMoment); // Berechnet DK's - eg V_i(t) falls
                                                                // lMoment = 1
  double         dGetCF(long lTime, long lInitState, long lTimeState); // Berechnet erwartete CF
                 // E[CF(t) x \chi_{I_t = lTimeState} | X(Stopzeit) = lInitState]
                 // Wenn man den total CF will muss man also
                 // summe_i dGetCF(long lTime, long lInitState, i) rechnen
  double         dGetRP(long lTime, long lState); // Berechnet Risikopraemie
  double         dGetSP(long lTime, long lState); // Berechnet Sparpraemie
  double         dGetRegP(long lTime, long lState); // Berechnet Regulaeren Zahlungsstrom
  long           lSetFolgezustand(long lStateVon, long lStateNach);
  long           lGetMaxTime();
  long           lGetNrStates();
  long           lGetStartTime();
  long           lGetStopTime();
  bool           dAddBenefits;
  void           vSetInitState(long lInitState);
  void           vGenerateTrajectory();
  long           vGetState(long lTime);
  double         dGetRandCF(long lTime);
  double         dGetRandDK(long lTime, long lMoment);
  double         dGetMeanCF(long lTime, long lState, long lNrSim);
  double         dGetMeanDK(long lTime, long lState, long lNrSim);
  void           vNewSeed(long lSeed);
  void           vResetMeanResults();
  long           lSeed;
  void vPrintTeX(FILE * psymTeXFile, bool bWithHeader, char * pcTitle, bool bAllEntries);
  TABLESERVER * psymTable1;
  TABLESERVER * psymTable2;
 private: // Hier folgend die internen Variablen, welche der obigen Nomeklatur folgen
  long           lMaxTimes;
  long           lMaxStates;
  long           lStartTime;
  long           lStopTime;
  long           lNrStates;
  long           lNrDefaultMoments;
  LV_VECTOR     *psymPre;   // Das sind die verketteten Vektoren
  LV_VECTOR     *psymPost;  // Man beachte dass der Zusammenhang und das Synchronisieren
  LV_VECTOR     *psymPij;   // ppsymPreInfo; pppsymPostInfo; pppsymPijInfo;pppsymDiscInfo;
  LV_VECTOR     *psymDisc;  // erfolgt, eg pppsymPostInfo[istate][jstate]->dGetValue(time)
  LV_MATRIX     *psymDK;    // Das sind die Internen Vektoren um Daten zu speichern DKs
```

```
  LV_MATRIX     *psymCF;    // Das sind die Internen Vektoren um Daten zu speichern CFs
  LV_VECTOR    **ppsymPreInfo;
  LV_VECTOR   ***pppsymPostInfo;
  LV_VECTOR   ***pppsymPijInfo;
  LV_VECTOR   ***pppsymDiscInfo;
  LV_MATRIX     *psymDKInfo[MAXMOMENTE]; // psymDKInfo[moment] -> dGetValue(time, state)
  LV_MATRIX    **ppsymCFInfo;            // ppsymCFInfo[lInitState]->dGetValue(lTime, lTimeState)
  long          *plFolgezustand;
  void           vPrepareInfoPointers(); // Hier werden die Datenstrukturen vorbereitet.
  long           lDKCalculated; // DKs berechnet ?
  long           lCFCalculated; // CF Berechnet ?
  long           lTechZerCalculated; // Technische Zerlegung berechnet ?
  bool           bGetData;
  bool           bStochasticInterest;
  ILV_VECTOR  * psymAktTraj;
  LV_VECTOR   * psymAktCF;
  LV_VECTOR   * psymAktDK;
  LV_VECTOR   * psymAktDisc;
  long         lNrTrajSim;
  long         lInitState;
  LV_MATRIX   * psymAggregCF;
  LV_MATRIX   * psymAggregDK;
};

#endif
```

Chapter 4

omarkov.cpp

```
//////////////////////////////////////////////////////////////////////
//                                                                  //
// Markovobjekt fuer LV Zahlungsstroeme                             //
//                                                                  //
// Autor Michael Koller                                             //
//                                                                  //
// Datum: March 2011: erstellt                                      //
//                                                                  //
//////////////////////////////////////////////////////////////////////

#include "omarkov.h"

const char * strPrgVersionStatic = "% This is omarkov V2.00 - Michael Koller 2011 \n";

// Wenn deterministischer Zins, so steht der Zins für alle Elemente auf der Diagonale

// DIE FOLGENDEN ZEILEN FUER INLINE CODE - eg Zufallszahlgenerator + Tex

#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

float ran2(long *idum)
{
  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  float temp;

  if (*idum <= 0) {
    if (-(*idum) < 1) *idum=1;
    else *idum = -(*idum);
    idum2=(*idum);
```

```
    for (j=NTAB+7;j>=0;j--) {
      k=(*idum)/IQ1;
      *idum=IA1*(*idum-k*IQ1)-k*IR1;
      if (*idum < 0) *idum += IM1;
      if (j < NTAB) iv[j] = *idum;
    }
    iy=iv[0];
  }
  k=(*idum)/IQ1;
  *idum=IA1*(*idum-k*IQ1)-k*IR1;
  if (*idum < 0) *idum += IM1;
  k=idum2/IQ2;
  idum2=IA2*(idum2-k*IQ2)-k*IR2;
  if (idum2 < 0) idum2 += IM2;
  j=iy/NDIV;
  iy=iv[j]-idum2;
  iv[j] = *idum;
  if (iy < 1) iy += IMM1;
  if ((temp=AM*iy) > RNMX) return RNMX;
  else return temp;
}
#undef IM1
#undef IM2
#undef AM
#undef IMM1
#undef IA1
#undef IA2
#undef IQ1
#undef IQ2
#undef IR1
#undef IR2
#undef NTAB
#undef NDIV
#undef EPS
#undef RNMX
/* (C) Copr. 1986-92 Numerical Recipes Software +1[L. */

void vPrintTexNumber(FILE* psymFile, double dNumber)
{
  char *pcTexFormat[]=
  {
    " & %10.6f",  // For small numbers
    " & %10.3f",  // for big numbers such as cash flows and mr
    " & %10.1f"   // for big numbers such as cash flows and mr
  };

    if (dNumber > 10000 || dNumber < -10000  ) fprintf(psymFile,pcTexFormat[2],dNumber);
    else{
         if (dNumber > 100 || dNumber < -100  ) fprintf(psymFile,pcTexFormat[1],dNumber);
         else fprintf(psymFile,pcTexFormat[0],dNumber);
         }
}


// Die folgende Klasse implementiert einen verketteten Leistungs oder W'keits-vektor

LV_VECTOR::LV_VECTOR(long lXDim, long lVonIpt, long lNachIpt)
{
  lXMax    = lXDim;
  pdValues = new double [lXMax];
  lVon     = lVonIpt;
  lNach    = lNachIpt;
  psymNext = NULL;
  memset(pdValues, 0, lXMax * sizeof(double));
}
```

```
LV_VECTOR::~LV_VECTOR()
{
  delete(pdValues);
}

void LV_VECTOR::vReset()
{
  memset(pdValues, 0, lXMax * sizeof(double));
}

double LV_VECTOR::dSetValue(long lX, double dValue)
{
  if(lX >= 0 && lX < lXMax)
    {
      pdValues[lX] = dValue;
      return(pdValues[lX]);
    }
  return(dValue - 1.);
}

double LV_VECTOR::dAddValue(long lX, double dValue)
{
  if(lX >= 0 && lX < lXMax)
    {
      pdValues[lX] += dValue;
      return(pdValues[lX]);
    }
  return(0.);
}

double LV_VECTOR::dGetValue(long lX)
{
  if(lX >= 0 && lX < lXMax)
    {
      return(pdValues[lX]);
    }
  return(0.);
}
////////////////////////////////////////////////////////////////////////////
// Die folgende Klasse implementiert einen verketteten Leistungs oder W'keits-vektor

ILV_VECTOR::ILV_VECTOR(long lXDim, long lVonIpt, long lNachIpt)
{
  lXMax    = lXDim;
  plValues = new long [lXMax];
  lVon     = lVonIpt;
  lNach    = lNachIpt;
  psymNext = NULL;
  memset(plValues, 0, lXMax * sizeof(long));
}

ILV_VECTOR::~ILV_VECTOR()
{
  delete(plValues);
}

void ILV_VECTOR::vReset()
{
  memset(plValues, 0, lXMax * sizeof(long));
}

long ILV_VECTOR::lSetValue(long lX, long lValue)
{
  if(lX >= 0 && lX < lXMax)
```

```
    {
      plValues[lX] = lValue;
      return(plValues[lX]);
    }
  return(lValue - 1);
}

long ILV_VECTOR::lAddValue(long lX, long lValue)
{
  if(lX >= 0 && lX < lXMax)
    {
      plValues[lX] += lValue;
      return(plValues[lX]);
    }
  return(0);
}

long ILV_VECTOR::lGetValue(long lX)
{
  if(lX >= 0 && lX < lXMax)
    {
      return(plValues[lX]);
    }
  return(0);
}
//////////////////////////////////////////////////////////////////////

LV_MATRIX::LV_MATRIX(long lXDim, long lYDim, long lIdentIpt)
{
  long lIc1;
  double * pdTemp;
  pdValues = new double [lXDim*lYDim];
  ppdValues = new double * [lXDim];
  pdTemp = pdValues;
  for(lIc1 = 0; lIc1 < lXDim; ++ lIc1)
    {
      ppdValues[lIc1] = pdTemp;
      pdTemp += lYDim;
    }
  lXMax    = lXDim;
  lYMax    = lYDim;
  lIdent   = lIdentIpt;
  psymNext = NULL;
  memset(pdValues, 0, lXMax * lYMax * sizeof(double));
}

LV_MATRIX::~LV_MATRIX()
{
  delete(pdValues);
  delete(ppdValues);
}

void LV_MATRIX::vReset()
{
  memset(pdValues, 0, lXMax * lYMax * sizeof(double));
}

double LV_MATRIX::dSetValue(long lX, long lY, double dValue)
{
  if(lX >= 0 && lX < lXMax && lY >= 0 && lY < lYMax  )
    {
      ppdValues[lX][lY] = dValue;
      return(ppdValues[lX][lY]);
    }
  return(dValue - 1.);
```

```
}

double LV_MATRIX::dAddValue(long lX, long lY, double dValue)
{
  if(lX >= 0 && lX < lXMax && lY >= 0 && lY < lYMax  )
    {
      ppdValues[lX][lY] += dValue;
      return(ppdValues[lX][lY]);
    }
  return(0.);
}

double LV_MATRIX::dMultiplyValue(long lX, long lY, double dValue)
{
  if(lX >= 0 && lX < lXMax && lY >= 0 && lY < lYMax  )
    {
      ppdValues[lX][lY] *= dValue;
      return(ppdValues[lX][lY]);
    }
  return(0.);
}

double LV_MATRIX::dGetValue(long lX, long lY)
{
  if(lX >= 0 && lX < lXMax && lY >= 0 && lY < lYMax  )
    {
      return(ppdValues[lX][lY]);
    }
  return(0.);
}


MARKOVLV::MARKOVLV(long lMaxTimesIpt, long lMaxStatesIpt, long lNrDefMomentsIpt)
{
  long lC1;
  LV_VECTOR ** psymVectTemp;

  // Im folgenden werden Alle defaults gesetzt. NULL bedeutet dass wir hier einen
  // Null pointer haben eg 0x0000. Das bedeutet, dass keine Elemente in der
  // entsprechenden Kette vorhanden sind und kein Memory alloziert
  lMaxTimes  = lMaxTimesIpt;
  lMaxStates = lMaxStatesIpt;
  lStartTime = 0l;
  lStopTime  = 0l;
  lNrStates  = 0l;
  lNrDefaultMoments = lNrDefMomentsIpt;
  psymPre    = NULL;
  psymPost   = NULL;
  psymPij    = NULL;
  psymDisc   = NULL;
  psymDK     = NULL;
  psymCF     = NULL;
  dAddBenefits = false;
  psymTable1 = NULL;
  psymTable2 = NULL;

#ifdef  DUMP_MARKOV_OBJ
  {
    FILE * psymDFObj;
    psymDFObj = fopen("c:\\omark.dat","w");
    fprintf(psymDFObj,"\n Objekt Markov: Dump");
  }
#endif

  // Die Idee der folgenden Pointers zB ppsymPreInfo ist auf das entsprechende Element in
```

```
  // den entsprechenden Datapools (in diesem Fall psymPre) zu zeigen um so einfach darauf
  // zugreifen zu koennen. Somit werden zuerst die entsprechenden Pointers angelegt und
  // mit NULL belegt, was heisst, dass dieser Vektor noch nicht existiert.
  // DIESE ARBEIT STARTET HIER ....
  ppsymPreInfo = new LV_VECTOR * [lMaxStates];
  for(lC1=0; lC1 < lMaxStates; ++ lC1)  ppsymPreInfo[lC1] = NULL;

  pppsymPostInfo = new LV_VECTOR ** [lMaxStates];
  psymVectTemp   = new LV_VECTOR *  [lMaxStates*lMaxStates];
  for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)  psymVectTemp[lC1] = NULL;
  for(lC1=0; lC1 < lMaxStates; ++ lC1)
    {
      pppsymPostInfo[lC1]  = psymVectTemp;
      psymVectTemp        += lMaxStates;
    }

  pppsymPijInfo  = new LV_VECTOR ** [lMaxStates];
  psymVectTemp   = new LV_VECTOR *  [lMaxStates*lMaxStates];
  for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)  psymVectTemp[lC1] = NULL;
  for(lC1=0; lC1 < lMaxStates; ++ lC1)
    {
      pppsymPijInfo[lC1]  = psymVectTemp;
      psymVectTemp        += lMaxStates;
    }


  pppsymDiscInfo = new LV_VECTOR ** [lMaxStates];
  psymVectTemp   = new LV_VECTOR *  [lMaxStates*lMaxStates];
  for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)  psymVectTemp[lC1] = NULL;
  for(lC1=0; lC1 < lMaxStates; ++ lC1)
    {
      pppsymDiscInfo[lC1]  = psymVectTemp;
      psymVectTemp        += lMaxStates;
    }

  ppsymCFInfo = new LV_MATRIX * [lMaxStates];
  for(lC1=0; lC1 < lMaxStates; ++ lC1) ppsymCFInfo[lC1] = NULL;

  // ... ENDET HIER

  lDKCalculated       = 0l; // Keine DKs berechnet
  lCFCalculated       = 0l; // Keine CFs berechnet
  bGetData            = false; // Wir schreiben Data
  bStochasticInterest = DEF_STOC_INT; // Normalerweise standart Zinsmodell
  for(lC1=0; lC1 < MAXMOMENTE; ++ lC1)
    psymDKInfo[lC1] = NULL; // Keine DK Pointers vorhanden

  // FOR risk premium
  lTechZerCalculated = 0; // Technische Zerlegung nicht berechnet
  plFolgezustand = new long [lMaxStates];
  for(lC1=0; lC1 < lMaxStates; ++ lC1) plFolgezustand[lC1] = lC1;

  // Fuer Simulation
  psymAktTraj = NULL;
  psymAktDisc = NULL;
  psymAktCF   = NULL;
  psymAktDK   = NULL;
  lSeed       = 12345671;
  lInitState  = 0l;
  lNrTrajSim  = 0l;
  psymAggregCF= NULL;
  psymAggregDK= NULL;
}
```

```
MARKOVLV::~MARKOVLV()
{
// Hier wird alles aufgeraeumt und die Daten frei gegeben
  LV_VECTOR * psymVectTmp, * psymOld;
  LV_MATRIX * psymMatrTmp, * psymMatrOld;

  // Wir gehen jeder Kette entlang und geben das entsprechende Memory frei
  // mit psymVectTmp = psymVectTmp->psymNext hangeln wir uns entlang der Kette
  // So lange bis wir auf das letzte Element kommen,
  // bei welchem psymVectTmp == NULL
  // DIESE ARBEIT STARTED HIER .....
  for(psymVectTmp = psymPre;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymVectTmp = psymPost;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymVectTmp = psymPij;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymVectTmp = psymDisc;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymMatrTmp = psymDK;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
    }

  for(psymMatrTmp = psymCF;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
    }
  // ... ENDET HIER, jetzt muessen wir noch die
  //                      Infopointers loeschen:
  delete(*pppsymPostInfo);
  delete(*pppsymPijInfo);
  delete(*pppsymDiscInfo);
  delete(pppsymPostInfo);
  delete(pppsymPijInfo);
  delete(pppsymDiscInfo);
  delete(ppsymPreInfo);
  delete(ppsymCFInfo);
  delete(plFolgezustand);
  if(psymAktTraj  != NULL) delete(psymAktTraj);
  if(psymAktCF    != NULL) delete(psymAktCF);
```

```
  if(psymAktDK    != NULL) delete(psymAktDK);
  if(psymAggregCF != NULL) delete(psymAggregCF);
  if(psymAggregDK != NULL) delete(psymAggregDK);
  if(psymAktDisc  != NULL) delete(psymAktDisc);
  if(psymTable1   != NULL) delete(psymTable1);
  if(psymTable2   != NULL) delete(psymTable2);

  lNrTrajSim     = 0l;
}

void MARKOVLV::vReset()
{ // Im wesentlichen werden die volatilen Teile der Daten wie beim
  // Destruktor geloescht - Kommentare she dort

  // Im Gegensatz zum Destruktor werden die Info-pointers nicht geloescht
  // und nur auf NULL (eg "keine Daten vorhanden") gesetzt

  LV_VECTOR * psymVectTmp, * psymOld;
  LV_MATRIX * psymMatrTmp, * psymMatrOld;
  long lC1;

  lDKCalculated =0l;
  lCFCalculated =0l;
  dAddBenefits = false;

  for(psymVectTmp = psymPre;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymVectTmp = psymPost;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymVectTmp = psymPij;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymVectTmp = psymDisc;psymVectTmp != NULL;)
    {
      psymOld = psymVectTmp;
      psymVectTmp = psymVectTmp->psymNext;
      delete(psymOld);
    }

  for(psymMatrTmp = psymDK;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
    }

  for(psymMatrTmp = psymCF;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
```

```
    }
  for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)
    {
      (*pppsymPostInfo)[lC1] = NULL;
      (*pppsymPijInfo)[lC1] = NULL;
      (*pppsymDiscInfo)[lC1] = NULL;
    }

  for(lC1=0; lC1 < lMaxStates; ++ lC1)
    {
      ppsymPreInfo[lC1] = NULL;
    }

  psymPre   = NULL;
  psymPost  = NULL;
  psymPij   = NULL;
  psymDisc  = NULL;
  psymDK    = NULL;
  psymCF    = NULL;

  for(lC1=0; lC1 < MAXMOMENTE; ++ lC1)
    psymDKInfo[lC1] = NULL;

  for(lC1=0; lC1 < lMaxStates; ++ lC1)
    ppsymCFInfo[lC1] = NULL;

  for(lC1=0; lC1 < lMaxStates; ++ lC1) plFolgezustand[lC1] = lC1;


}

void MARKOVLV::vSetInternals(long lMaxTimesInput, long lMaxStatesInput)
{
  long lNrDefMomentsIpt = lNrDefaultMoments;
  // Zuerst Löschen wir alles umd dann alles wieder aufzusetzen wie beim
  // Konstruktor
  {
    LV_VECTOR * psymVectTmp, * psymOld;
    LV_MATRIX * psymMatrTmp, * psymMatrOld;

    this->vReset(); // Dies loescht alles bei vorgegebener Sturktur
    // da hier aber die beiden 'Internals' neu gesetzt werden
    // somit muessen die untenstehenden Daten zuerst geloescht und dann
    // wieder aufgesetzt werden.

    for(psymVectTmp = psymPre;psymVectTmp != NULL;)
      {
        psymOld = psymVectTmp;
        psymVectTmp = psymVectTmp->psymNext;
        delete(psymOld);
      }

    for(psymVectTmp = psymPost;psymVectTmp != NULL;)
      {
        psymOld = psymVectTmp;
        psymVectTmp = psymVectTmp->psymNext;
        delete(psymOld);
      }

    for(psymVectTmp = psymPij;psymVectTmp != NULL;)
      {
        psymOld = psymVectTmp;
        psymVectTmp = psymVectTmp->psymNext;
        delete(psymOld);
      }
```

```
    for(psymVectTmp = psymDisc;psymVectTmp != NULL;)
      {
        psymOld = psymVectTmp;
        psymVectTmp = psymVectTmp->psymNext;
        delete(psymOld);
      }

    for(psymMatrTmp = psymDK;psymMatrTmp != NULL;)
      {
        psymMatrOld = psymMatrTmp;
        psymMatrTmp = psymMatrTmp->psymNext;
        delete(psymMatrOld);
      }

    for(psymMatrTmp = psymCF;psymMatrTmp != NULL;)
      {
        psymMatrOld = psymMatrTmp;
        psymMatrTmp = psymMatrTmp->psymNext;
        delete(psymMatrOld);
      }

    delete(*pppsymPostInfo);
    delete(*pppsymPijInfo);
    delete(*pppsymDiscInfo);
    delete(pppsymPostInfo);
    delete(pppsymPijInfo);
    delete(pppsymDiscInfo);
    delete(ppsymPreInfo);
    delete(ppsymCFInfo);
    delete(plFolgezustand);
  } // Jetzt ist alles geloescht - wie beim Destruktor
  { // Und jetzt folgt derselbe Code fuer den Konstruktor mit
    // den neuen Parametern
    long lC1;
    LV_VECTOR ** psymVectTemp;

    lMaxTimes  = lMaxTimesInput;
    lMaxStates = lMaxStatesInput;
    lStartTime = 0l;
    lStopTime  = 0l;
    lNrStates  = 0l;
    lNrDefaultMoments = lNrDefMomentsIpt;
    psymPre    = NULL;
    psymPost   = NULL;
    psymPij    = NULL;
    psymDisc   = NULL;
    psymDK     = NULL;
    psymCF     = NULL;
    dAddBenefits = false;

    ppsymPreInfo = new LV_VECTOR * [lMaxStates];
    for(lC1=0; lC1 < lMaxStates; ++ lC1)  ppsymPreInfo[lC1] = NULL;

    pppsymPostInfo = new LV_VECTOR ** [lMaxStates];
    psymVectTemp   = new LV_VECTOR *  [lMaxStates*lMaxStates];
    for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)  psymVectTemp[lC1] = NULL;
    for(lC1=0; lC1 < lMaxStates; ++ lC1)
      {
        pppsymPostInfo[lC1]  = psymVectTemp;
        psymVectTemp        += lMaxStates;
      }

    pppsymPijInfo  = new LV_VECTOR ** [lMaxStates];
    psymVectTemp   = new LV_VECTOR *  [lMaxStates*lMaxStates];
```

```
    for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)  psymVectTemp[lC1] = NULL;
    for(lC1=0; lC1 < lMaxStates; ++ lC1)
      {
        pppsymPijInfo[lC1]  = psymVectTemp;
        psymVectTemp        += lMaxStates;
      }


    pppsymDiscInfo = new LV_VECTOR ** [lMaxStates];
    psymVectTemp   = new LV_VECTOR *  [lMaxStates*lMaxStates];
    for(lC1=0; lC1 < lMaxStates*lMaxStates; ++ lC1)  psymVectTemp[lC1] = NULL;
    for(lC1=0; lC1 < lMaxStates; ++ lC1)
      {
        pppsymDiscInfo[lC1]  = psymVectTemp;
        psymVectTemp        += lMaxStates;
      }

    ppsymCFInfo = new LV_MATRIX * [lMaxStates];
    for(lC1=0; lC1 < lMaxStates; ++ lC1) ppsymCFInfo[lC1] = NULL;

    lDKCalculated       = 0l;
    lCFCalculated       = 0l;
    bGetData            = false;
    bStochasticInterest = DEF_STOC_INT;
    for(lC1=0; lC1 < MAXMOMENTE; ++ lC1)
      psymDKInfo[lC1] = NULL;

    lTechZerCalculated = 0;
    plFolgezustand = new long [lMaxStates];
    for(lC1=0; lC1 < lMaxStates; ++ lC1) plFolgezustand[lC1] = lC1;
  }
  if(psymAktTraj  != NULL) delete(psymAktTraj);
  if(psymAktCF    != NULL) delete(psymAktCF);
  if(psymAktDK    != NULL) delete(psymAktDK);
  if(psymAggregCF != NULL) delete(psymAggregCF);
  if(psymAggregDK != NULL) delete(psymAggregDK);
  if(psymAktDisc  != NULL) delete(psymAktDisc);
  lNrTrajSim     = 0l;
  psymAktTraj = NULL;
  psymAktDisc = NULL;
  psymAktCF   = NULL;
  psymAktDK   = NULL;
  lSeed       = 12345671;
  lInitState  = 0l;
  lNrTrajSim  = 0l;
  psymAggregCF= NULL;
  psymAggregDK= NULL;
}

void MARKOVLV::vSetStartTime(long lTime)
{ // Hier wird die Startzeit gesetzt
  LV_MATRIX * psymMatrTmp;
  lStartTime = lTime;
  if(lTime <0) lStartTime = 0; // Plausibility checks
  if(lTime >= lMaxTimes) lStartTime = lMaxTimes-1;
  if(lDKCalculated > 0) lDKCalculated = -lDKCalculated; // wenn berechnet, muss das neu
  // berechnet werden. Dies bedeutet diese negative Zahl
  for(psymMatrTmp = psymDK;psymMatrTmp != NULL;psymMatrTmp = psymMatrTmp->psymNext)
    psymMatrTmp->vReset(); // Da die DKs nicht mehr gueltig sind, werden die auf Null gesetzt
  if(lCFCalculated > 0) lCFCalculated = -lCFCalculated; // Dto fuer Cash flows
  for(psymMatrTmp = psymCF;psymMatrTmp != NULL;psymMatrTmp = psymMatrTmp->psymNext)
    psymMatrTmp->vReset(); // auch hier CF auf Null gesetzt
}
void MARKOVLV::vSetStopTime(long lTime)
{ // Hier wird die Stopzeit neu gesetzt
```

```
  LV_MATRIX * psymMatrTmp;
  lStopTime = lTime;
  if(lTime <0) lStopTime = 0;
  if(lTime >= lMaxTimes) lStopTime = lMaxTimes-1;
  if(lDKCalculated > 0) lDKCalculated = -lDKCalculated; // Wie oben Berechnung ungueltig
  // und zuruecksetzten der Werte auf Null
  for(psymMatrTmp = psymDK;psymMatrTmp != NULL;psymMatrTmp = psymMatrTmp->psymNext)
    psymMatrTmp->vReset();
  if(lCFCalculated > 0) lCFCalculated = -lCFCalculated;
  for(psymMatrTmp = psymCF;psymMatrTmp != NULL;psymMatrTmp = psymMatrTmp->psymNext)
    psymMatrTmp->vReset();
}

void MARKOVLV::vSetNrStates(long lNrStatesIpt)
{ // Setzten der Anzahl Zustaende
  long lC1;
  LV_MATRIX * psymMatrTmp, * psymMatrOld;
  lNrStates = lNrStatesIpt;
  if(lNrStatesIpt <0) lNrStates = 1;
  if(lNrStatesIpt >= lMaxStates) lNrStates = lMaxStates; // Hier muss man aufpassen!!
  // Wenn das Objekt zB mit max 5 zustaenden kreiert wurde kann man hier nicht 8 zustaende
  // wollen. Das prg setzt auf das max - im Bsp 5
  lDKCalculated = 0; // Jetzt wird alles zurueckgesetzt - im Gegensatz zu dem aendern der Zeiten
  lCFCalculated = 0; // Im Folgenden alle Datenstrukturen loeschen.
  for(psymMatrTmp = psymDK;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
    }

  for(psymMatrTmp = psymCF;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
    }
  for(lC1=0; lC1 < MAXMOMENTE; ++ lC1)
    psymDKInfo[lC1] = NULL;

}
void   MARKOVLV::vSetGetData(bool bStatus)
{
  bGetData = bStatus;
}

double MARKOVLV::dSetPre(long lTime, long lVon, long lNach, double dValue)
{ // Hier wird a_i(t) gesetzt, danach alle andern.
  // Das interessante ist die Datenstruktur. Falls bereits ein Vektor angelegt ist,
  // so muss der wert nur gesetzt werden (CC). andernfalls muss am ende der entsprechenden
  // kette ein neuer vektor (AA) angelegt werden. Hierfuer muss der letzte wert gefunden
  // werden (BB)
  LV_VECTOR * psymTemp;
  if(bGetData)
    {
      if(ppsymPreInfo[lVon] == NULL)
        {
          return(0.);
        }
      return(ppsymPreInfo[lVon]->dGetValue(lTime));
    }

  if(ppsymPreInfo[lVon] == NULL) // Wir sind in Fall AA und suchen BB
    {
      if(psymPre == NULL) // erster wert BB gefunden
```

```
        {
          psymPre = new LV_VECTOR(lMaxTimes, lVon, lVon); // neuer vektor AA
          ppsymPreInfo[lVon] = psymPre; // wo sitzt er
        }
      else
        {
          for(psymTemp=psymPre; psymTemp->psymNext != NULL;) // wir suchen letzten wert
            psymTemp =psymTemp -> psymNext;
          psymTemp -> psymNext = new LV_VECTOR(lMaxTimes, lVon, lVon); // gefunden BB und setzen AA
          ppsymPreInfo[lVon] = psymTemp -> psymNext;

        }
    }
  if (dAddBenefits) return(ppsymPreInfo[lVon]->dAddValue(lTime, dValue)); // Jetzt setzen wird Wert CC
  else return(ppsymPreInfo[lVon]->dSetValue(lTime, dValue)); // CC
}

double MARKOVLV::dSetPost(long lTime, long lVon, long lNach, double dValue)
{ // Genau wie oben setze nur Marker AA, BB, CC
  LV_VECTOR * psymTemp;
  if(bGetData)
    {
      if(pppsymPostInfo[lVon][lNach] == NULL)
        {
          return(0.);
        }
      return(pppsymPostInfo[lVon][lNach]->dGetValue(lTime));
    }

  if(pppsymPostInfo[lVon][lNach] == NULL)
    {
      if(psymPost == NULL)  // BB
        {
          psymPost = new LV_VECTOR(lMaxTimes, lVon, lNach); // AA
          pppsymPostInfo[lVon][lNach] = psymPost; // und setzen info
        }
      else
        {
          for(psymTemp=psymPost; psymTemp->psymNext != NULL;) // BB
            psymTemp =psymTemp -> psymNext;
          psymTemp -> psymNext = new LV_VECTOR(lMaxTimes, lVon, lNach); // AA
          pppsymPostInfo[lVon][lNach] = psymTemp -> psymNext; // und setzen info

        }
    }
  if (dAddBenefits) return(pppsymPostInfo[lVon][lNach]->dAddValue(lTime, dValue));
  else return(pppsymPostInfo[lVon][lNach]->dSetValue(lTime, dValue)); // CC
}


double MARKOVLV::dSetPij(long lTime, long lVon, long lNach, double dValue)
{ // ANALOG post
  LV_VECTOR * psymTemp;
  if(bGetData)
    {
      if(pppsymPijInfo[lVon][lNach] == NULL)
        {
          return(0.);
        }
      return(pppsymPijInfo[lVon][lNach]->dGetValue(lTime));
    }

  if(pppsymPijInfo[lVon][lNach] == NULL)
    {
      if(psymPij == NULL)
```

```
        {
          psymPij = new LV_VECTOR(lMaxTimes, lVon, lNach);
          pppsymPijInfo[lVon][lNach] = psymPij;
        }
      else
        {
          for(psymTemp=psymPij; psymTemp->psymNext != NULL;)
            psymTemp =psymTemp -> psymNext;
          psymTemp -> psymNext = new LV_VECTOR(lMaxTimes, lVon, lNach);
          pppsymPijInfo[lVon][lNach] = psymTemp -> psymNext;

        }
    }
  return(pppsymPijInfo[lVon][lNach]->dSetValue(lTime, dValue));
}


double MARKOVLV::dSetDisc(long lTime, long lVon, long lNach, double dValue)
{ // analog post
  LV_VECTOR * psymTemp;
  if(bStochasticInterest == false) lNach = lVon;
  if(bGetData)
    {
      if(pppsymDiscInfo[lVon][lNach] == NULL)
        {
          return(0.);
        }
      return(pppsymDiscInfo[lVon][lNach]->dGetValue(lTime));
    }

  if(pppsymDiscInfo[lVon][lNach] == NULL)
    {
      if(psymDisc == NULL)
        {
          psymDisc = new LV_VECTOR(lMaxTimes, lVon, lNach);
          pppsymDiscInfo[lVon][lNach] = psymDisc;
        }
      else
        {
          for(psymTemp=psymDisc; psymTemp->psymNext != NULL;)
            psymTemp =psymTemp -> psymNext;
          psymTemp -> psymNext = new LV_VECTOR(lMaxTimes, lVon, lNach);
          pppsymDiscInfo[lVon][lNach] = psymTemp -> psymNext;
        }
    }
  return(pppsymDiscInfo[lVon][lNach]->dSetValue(lTime, dValue));
}


void MARKOVLV::vSetInterestModel(bool bStocInterest)
{
  if(bStochasticInterest != bStocInterest) vReset();
  bStochasticInterest = bStocInterest;
}

void MARKOVLV::vSetDefaultNrMoments(long lNrMoments)
{ // wie beim aendern der Zeiten etc, selbes vorgehen
  LV_MATRIX * psymMatrTmp, * psymMatrOld;
  long lC1;

  for(psymMatrTmp = psymDK;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
```

```
    }

  for(psymMatrTmp = psymCF;psymMatrTmp != NULL;)
    {
      psymMatrOld = psymMatrTmp;
      psymMatrTmp = psymMatrTmp->psymNext;
      delete(psymMatrOld);
    }

  for(lC1=0; lC1 < MAXMOMENTE; ++ lC1)
    psymDKInfo[lC1] = NULL;

  for(lC1=0; lC1 < lMaxStates; ++ lC1)
    ppsymCFInfo[lC1] = NULL;

  lDKCalculated =0l;
  lCFCalculated =0l;

  lNrDefaultMoments = lNrMoments;
}

double MARKOVLV::dGetDK(long lTime, long lState, long lMoment)
{
  // Hier wird wirklich etwas gerechnet. Es gibt zwei Faelle. Wenn es berechnt ist
  // muss nur der Wert zurueckgegeben werden, sonst rechnen
  int lC1;
  LV_MATRIX * psymTempMatr;
  if(lMoment <= 0) lMoment = 1;
  if(lDKCalculated >= lMoment) // Alle Werte bis zum lDKCalculated Moment berechnet
    {                          // ... gut und wir koennen nur den Wert zurueckgeben
      return(psymDKInfo[lMoment]->dGetValue(lTime, lState));
    }
  if(lDKCalculated == 0) // In diesem Fall muessen wir rechnen
    { // Zuerst muessen wir fuer jedes Moment eine matrix zeit x zustand
      // bereitstellen ... START
      psymDK = new LV_MATRIX(lMaxTimes, lNrStates, 1l);
      psymTempMatr  = psymDK;
      psymDKInfo[1] = psymDK;

      for(lC1=2; lC1<= lNrDefaultMoments; ++lC1)
        {
          psymTempMatr->psymNext = new LV_MATRIX(lMaxTimes, lNrStates, lC1);
          psymDKInfo[lC1] = psymTempMatr->psymNext;
          psymTempMatr    = psymTempMatr->psymNext;
        }
    } // .. END DATEN BEREITSTELLEN
  {   // Jetzt beginnen wir zu rechnen
    /* ------------------------------------------------------------ */
    /* 0. Variabeln definieren und belegen                          */
    /* ------------------------------------------------------------ */
    long   laufzeit;
    long   itemp, jtemp, kstate;
    long   istate, jstate;   /* zustand i -> zustand j              */
    double akt_disk;         /* diskont fuer aktuelle periode       */
    double p_ij_t;           /* uebergangswahrscheinlichkei  t      */
    double l_ij_tv, l_ij_tn; /* leistung die beim uebergang faeloig wird */
    double dBin[MAXMOMENTE][MAXMOMENTE];
    long   lOrdn;
    LV_VECTOR * psymTemp;

    /* ------------------------------------------------------------ */
    /* 1. Randbedingung belegen und relev Uebergaenge bestimmen     */
    /* ------------------------------------------------------------ */
    for(itemp=0;itemp<MAXMOMENTE;++itemp)
      {
```

```
      memset(dBin[itemp],0,MAXMOMENTE*sizeof(double));
    }

  for(itemp=0;itemp<=lNrDefaultMoments;++itemp)
    {
      for(jtemp=0; jtemp<=itemp;++jtemp)
        {
          if(jtemp == 0 || jtemp == itemp)
            {
              dBin[itemp][jtemp] =1.;
            }
          else
            {
              dBin[itemp][jtemp] = dBin[itemp-1][jtemp-1] + dBin[itemp-1][jtemp];
            }
        }
    }



  /* ------------------------------------------------------------ */
  /* 2. Eigentliche Rechnung                                      */
  /* ------------------------------------------------------------ */
  /* an dieser stelle soll die verwendte formel erklaert werden   */
  /* mit Vj bezeichne man die reserve                             */
  /* dann gilt :                                                  */
  /* Vj(t) = summe(g) [ pjg(t,s)(z(j,g) + eing-disk * Vg(s)]      */
  /* wobei z(j,g) die faellige Leist bezeichnet                   */
  /* ------------------------------------------------------------ */
  for(laufzeit = lStartTime - 1; laufzeit >= lStopTime; --laufzeit)
    {

      for(lOrdn=1;lOrdn<=lNrDefaultMoments;++lOrdn)
        {

          for(psymTemp = psymPij;psymTemp != NULL; psymTemp = psymTemp->psymNext)

            {
              istate = psymTemp->lVon;
              jstate = psymTemp->lNach;

              if(bStochasticInterest) kstate = jstate;
              else                    kstate = istate;

              if(pppsymDiscInfo[istate][kstate] != NULL)
                akt_disk = pppsymDiscInfo[istate][kstate]->dGetValue(laufzeit);
                                          /* aktueller diskont fuer 1 periode */
              else
                akt_disk = 0.;

              /* uebergangsw'keit */
              p_ij_t =psymTemp->dGetValue(laufzeit);
              /* dabei faellige leistung */
              if(pppsymPostInfo[istate][jstate] != NULL)
                l_ij_tn =pppsymPostInfo[istate][jstate]->dGetValue(laufzeit);
              else
                l_ij_tn = 0.;
              {
                double dTeil = 0.;
                double dFakt = 1.;
                int    iLocC;
                for(iLocC=0; iLocC<lOrdn;++iLocC)
                  {
        dTeil  += dBin[lOrdn][iLocC]*dFakt*psymDKInfo[lOrdn-iLocC]->dGetValue(laufzeit+1,jstate);
        dFakt  *= l_ij_tn;
```

```
                    }
                  dTeil  += dFakt;
                  psymDKInfo[lOrdn]->dAddValue(laufzeit,istate, p_ij_t* pow(akt_disk,lOrdn) * dTeil);
                }
                /* veraenderung der reserve */
                //l_ij_tn + *(m->reserve[jstate]+laufzeit + 1)));

              }
            /*ie.: reserve +=        leistv+w'keit *(  diskont  *(leistn + reserve )) */
          }
        for(istate = 0; istate < lNrStates; ++istate)
          {
            double dOld[MAXMOMENTE];
            memset(dOld, 0, MAXMOMENTE * sizeof(double));
            if(ppsymPreInfo[istate] != NULL)
              l_ij_tv = ppsymPreInfo[istate]->dGetValue(laufzeit);
            else
              l_ij_tv = 0.;
            for(lOrdn=1;lOrdn<=lNrDefaultMoments;++lOrdn)
              {
                dOld[lOrdn] =   psymDKInfo[lOrdn]->dGetValue(laufzeit,istate);
              }

            for(lOrdn=1;lOrdn<=lNrDefaultMoments;++lOrdn)
              {
                double dTeil = 0.;
                double dFakt = 1.;
                int    iLocC;

                for(iLocC=0; iLocC<lOrdn;++iLocC)
                  {
                    dTeil  += dBin[lOrdn][iLocC] * dFakt *  dOld[lOrdn-iLocC];
                    dFakt  *= l_ij_tv;
                  }
                dTeil  += dFakt;
                psymDKInfo[lOrdn]->dSetValue(laufzeit,istate,dTeil);
              }
          }
      }
  }

  lDKCalculated = lNrDefaultMoments;  // Jetzt sagen wir dem objekt dass wir berechnet
                                      // laben, was wir wollten
  if(lMoment > lDKCalculated) return(0.);
  return(psymDKInfo[lMoment]->dGetValue(lTime, lState)); // geben resultat zurueck.
}

double MARKOVLV::dGetCF(long lTime, long lInitState, long lTimeState)
{
  int lC1;
  LV_MATRIX * psymTempMatr;
  if(lInitState >= lNrStates || lInitState < 0 || lTimeState >= lNrStates || lTimeState <0) return(0.);
  if(lCFCalculated > 0)
    {
      return(ppsymCFInfo[lInitState]->dGetValue(lTime, lTimeState));
    }
  if(lCFCalculated == 0)
    {
      psymCF = new LV_MATRIX(lMaxTimes, lNrStates, 1l);
      psymTempMatr  = psymCF;
      ppsymCFInfo[0] = psymCF;

      for(lC1=1; lC1< lNrStates; ++lC1)
        {
          psymTempMatr->psymNext = new LV_MATRIX(lMaxTimes, lNrStates, lC1);
```

```
          ppsymCFInfo[lC1] = psymTempMatr->psymNext;
          psymTempMatr     = psymTempMatr->psymNext;
        }
    }
  /* Calculation of CF -- Start */
  {
    /* ------------------------------------------------------------ */
    /* 0. Variabeln definieren und belegen                          */
    /* ------------------------------------------------------------ */
    long   laufzeit;
    long   istate, jstate, kstate;   /* zustand i -> zustand j               */
    long   lCStartState;
    double akt_disk;         /* diskont fuer aktuelle periode        */
    double p_ij_t;           /* uebergangswahrscheinlichkei  t       */
    double l_ij_tv, l_ij_tn; /* leistung die beim uebergang faeloig wird */
    LV_VECTOR * psymTemp;
    double * pdPNext;
    double * pdPJetzt;

    pdPJetzt = new double [lNrStates];
    pdPNext  = new double [lNrStates];

    for(lCStartState = 0; lCStartState < lNrStates; ++ lCStartState)
      {
        memset(pdPJetzt, 0, lNrStates * sizeof(double));
        pdPJetzt[lCStartState] = 1;

        /* ------------------------------------------------------------ */
        /* 2. Eigentliche Rechnung                                      */
        /* ------------------------------------------------------------ */
        /* an dieser stelle soll die verwendte formel erklaert werden   */
        /* mit Vj bezeichne man die reserve                             */
        /* dann gilt :                                                  */
        /* Vj(t) = summe(g) [ pjg(t,s)(z(j,g) + eing-disk * Vg(s)]      */
        /* wobei z(j,g) die faellige Leist bezeichnet                   */
        for(laufzeit = lStopTime ; laufzeit < lStartTime; ++laufzeit)
          {
            memset(pdPNext, 0, lNrStates * sizeof(double));

            for(psymTemp = psymPij;psymTemp != NULL; psymTemp = psymTemp->psymNext)
              {
                istate = psymTemp->lVon;
                jstate = psymTemp->lNach;

                if(bStochasticInterest) kstate = jstate;
                else                    kstate = istate;

                if(pppsymDiscInfo[istate][kstate] != NULL)
                  akt_disk = pppsymDiscInfo[istate][kstate]->dGetValue(laufzeit);
                                        /* aktueller diskont fuer 1 periode */
                else
                  akt_disk = 0.;
                /* uebergangsw'keit */
                p_ij_t =psymTemp->dGetValue(laufzeit);
                pdPNext[jstate] += p_ij_t * pdPJetzt[istate];
                /* dabei faellige leistung */
                if(pppsymPostInfo[istate][jstate] != NULL)
                  l_ij_tn =pppsymPostInfo[istate][jstate]->dGetValue(laufzeit);
                else
                  l_ij_tn = 0.;

#ifndef ALLOCATE_CF_TO_EXACT_TIMES
#pragma message (">>>> CF zu Zeit des beginns der Periode")
          ppsymCFInfo[lCStartState]->dAddValue(laufzeit, istate, p_ij_t*( akt_disk * (l_ij_tn )));
#else
```

```
#pragma message (">>>> CF zur exakte Zeit")
          ppsymCFInfo[lCStartState]->dAddValue(laufzeit+1, jstate, p_ij_t* l_ij_tn );
#endif
              }
            for(istate=0; istate<lNrStates; ++istate)
              {
                if(ppsymPreInfo[istate] != NULL)
                  l_ij_tv = ppsymPreInfo[istate]->dGetValue(laufzeit);
                else
                  l_ij_tv = 0.;
                ppsymCFInfo[lCStartState]->dAddValue(laufzeit, istate, l_ij_tv);
              }

            /* 2a Überschieben mit Randverteilung */
            for(istate = 0; istate < lNrStates; ++istate)
              ppsymCFInfo[lCStartState]->dMultiplyValue(laufzeit,istate, pdPJetzt[istate]);
            for(istate = 0; istate < lNrStates; ++istate)
              pdPJetzt[istate] = pdPNext[istate] ;
          }
      }
    delete(pdPJetzt);
    delete(pdPNext);
  }
  /* Calculation of CF -- End */
  lCFCalculated = lNrStates;
  return(ppsymCFInfo[lInitState]->dGetValue(lTime, lTimeState));
}

long     MARKOVLV::lGetMaxTime(){return(lMaxTimes);}
long     MARKOVLV::lGetNrStates(){return(lNrStates);}
long     MARKOVLV::lGetStartTime(){return(lStartTime);}
long     MARKOVLV::lGetStopTime(){return(lStopTime);}

double         MARKOVLV::dGetRP(long lTime, long lState) // Berechnet Risikopraemie TODO
{
  long lNachNicht   = plFolgezustand[lState], kstate, lNach;
  double akt_disk, dWert= 0, dPart, dGeplant;
  /* Nun abziehen des normalen Zustandes: funktional analog aber mit - */
  {
    lNach    = lNachNicht;
    dGeplant = dGetDK(lTime+1, lNach, 1l);
    if(pppsymPostInfo[lState][lNach] != NULL)
      dGeplant += pppsymPostInfo[lState][lNach]->dGetValue(lTime+1);
  }

  for(lNach=0l; lNach < lNachNicht; ++lNach)
    {
      if(bStochasticInterest) kstate = lNach;
      else                    kstate = lState;

      if(pppsymDiscInfo[lState][kstate] != NULL)
        akt_disk = pppsymDiscInfo[lState][kstate]->dGetValue(lTime);
                                    /* aktueller diskont fuer 1 periode */
      else
        akt_disk = 0.;
      if(pppsymPijInfo[lState][lNach] != NULL)
        {
          akt_disk = pppsymDiscInfo[lState][kstate]->dGetValue(lTime);
          dPart = dGetDK(lTime+1, lNach, 1l) - dGeplant;
          if(pppsymPostInfo[lState][lNach] != NULL)
                         dPart += pppsymPostInfo[lState][lNach]->dGetValue(lTime+1);
          dWert += akt_disk * pppsymPijInfo[lState][lNach]->dGetValue(lTime) * dPart;
        }
    }
  for(lNach=lNachNicht+1; lNach < lNrStates; ++lNach)
```

```
    {
      if(bStochasticInterest) kstate = lNach;
      else                    kstate = lState;

      if(pppsymDiscInfo[lState][kstate] != NULL)
        akt_disk = pppsymDiscInfo[lState][kstate]->dGetValue(lTime);
                                              /* aktueller diskont fuer 1 periode */
      else
        akt_disk = 0.;
      if(pppsymPijInfo[lState][lNach] != NULL)
        {
          akt_disk = pppsymDiscInfo[lState][kstate]->dGetValue(lTime);
          dPart = dGetDK(lTime+1, lNach, 1l) - dGeplant;
          if(pppsymPostInfo[lState][lNach] != NULL)
                        dPart += pppsymPostInfo[lState][lNach]->dGetValue(lTime+1);
          dWert += akt_disk * pppsymPijInfo[lState][lNach]->dGetValue(lTime) * dPart;
        }
    }
  return(dWert);
}

double         MARKOVLV::dGetSP(long lTime, long lState) // Berechnet Sparpraemie
{
  long lNach   = plFolgezustand[lState], kstate;
  double akt_disk;

  if(bStochasticInterest) kstate = lNach;
  else                    kstate = lState;

  if(pppsymDiscInfo[lState][kstate] != NULL)
    akt_disk = pppsymDiscInfo[lState][kstate]->dGetValue(lTime);
                                          /* aktueller diskont fuer 1 periode */
  else
    akt_disk = 0.;
  return(akt_disk * dGetDK(lTime+1, lNach, 1l) - dGetDK(lTime, lState, 1l));
}

double         MARKOVLV::dGetRegP(long lTime, long lState) // Berechnet Regulaeren Zahlungsstrom
{
  double dWert = 0;
  long lNach   = plFolgezustand[lState];
  if(ppsymPreInfo[lState]) dWert = ppsymPreInfo[lState]->dGetValue(lTime);
  if(pppsymPostInfo[lState][lNach]->dGetValue(lTime))
                              dWert += pppsymPostInfo[lState][lNach]->dGetValue(lTime);
  return(dWert);
}

long           MARKOVLV::lSetFolgezustand(long lStateVon, long lStateNach)
{
  if(!bGetData)
    {
      plFolgezustand[lStateVon] = lStateNach;
      lTechZerCalculated = 0;
    }
  return(plFolgezustand[lStateVon]);
}

void           MARKOVLV::vSetInitState(long lInitState)
{
  this->lInitState = lInitState;
}

void          MARKOVLV::vGenerateTrajectory()
{
  /* Das ist die eigentliche Simualtionsroutine, welche hier sehr einfach implementiert ist. */
```

```
  long lTime;
  long lLastState;
  long lAktState, kstate;
  bool bExistTransition = false;
  double dSumProb, dRand;
  double l_ij_tv, l_ij_tn, akt_disk, dCF;
  double dDK;

  if (psymAktTraj == NULL) psymAktTraj = new ILV_VECTOR(lMaxTimes, 0, 0);
  if (psymAktCF   == NULL) psymAktCF   = new LV_VECTOR(lMaxTimes, 0, 0);;
  if (psymAktDK   == NULL) psymAktDK   = new LV_VECTOR(lMaxTimes, 0, 0);;
  if (psymAktDisc == NULL) psymAktDisc = new LV_VECTOR(lMaxTimes, 0, 0);;
  if (psymAggregCF== NULL) psymAggregCF= new LV_MATRIX(lMaxTimes, lNrStates, 0);
  if (psymAggregDK== NULL) psymAggregDK= new LV_MATRIX(lMaxTimes, lNrStates, 0);

  if(lStopTime <0) lStopTime = 0;
  if(lStopTime >= lMaxTimes) lStopTime = lMaxTimes-1;


  psymAktTraj->vReset();
  ++lNrTrajSim;
  lLastState  = lInitState;
  //printf("\n --> %ld ", lSeed);
  for (lTime = lStopTime; lTime < lStartTime; ++lTime)
    {
      dRand = ran2(&lSeed);
      dSumProb = 0.;
      bExistTransition = false;

      {
        for(lAktState=0; lAktState < lNrStates; ++ lAktState)
          {
            if(pppsymPijInfo[lLastState][lAktState] != NULL)
              {
                bExistTransition = true;
                dSumProb += pppsymPijInfo[lLastState][lAktState]->dGetValue(lTime);
              }
            if (dSumProb > dRand)
              {
                goto a;
              }
          }
        lAktState = lNrStates-1;
      }
    a:   psymAktTraj->lSetValue(lTime, lLastState);

      //printf("\n %d -> %d %d", lTime, lAktState, lLastState);
      if(!bExistTransition)
        lAktState = lLastState;
      if(bStochasticInterest) kstate = lAktState;
      else                    kstate = lLastState;
      // Berechnen Zins t -> t + 1
      if(pppsymDiscInfo[lLastState][kstate] != NULL)
        akt_disk = pppsymDiscInfo[lLastState][kstate]->dGetValue(lTime);   /* aktueller diskont fuer 1
      else
        akt_disk = 0.;
      psymAktDisc->dSetValue(lTime, akt_disk);
      // Berechnen Cash Flow per Anfang der Zeit t
      if(pppsymPostInfo[lLastState][lAktState] != NULL)
        l_ij_tn =pppsymPostInfo[lLastState][lAktState]->dGetValue(lTime);
      else
        l_ij_tn = 0.;
      if(ppsymPreInfo[lLastState] != NULL)
        l_ij_tv = ppsymPreInfo[lLastState]->dGetValue(lTime);
      else
```

```
        l_ij_tv = 0.;

      dCF =l_ij_tv + akt_disk * l_ij_tn;
      psymAktCF->dSetValue(lTime, dCF);
      psymAggregCF->dAddValue(lTime,lLastState, dCF);
      // Zeit Fortschreiben
      lLastState = lAktState;
    }
  dDK = 0;
  for (lTime = lStartTime-1; lTime >= lStopTime; --lTime)
    {
      dDK = dDK * psymAktDisc->dGetValue(lTime) + psymAktCF->dGetValue(lTime);
      //printf("\n %d Disk %f CF %f DK %f",lTime, psymAktDisc->dGetValue(lTime), psymAktCF->dGetValue(l
      psymAktDK->dSetValue(lTime,dDK);
      psymAggregDK->dAddValue(lTime,psymAktTraj->lGetValue(lTime),dDK);
    }
  //printf("\n lTime %d DK %f State %d TotDK %f", lTime, dDK, psymAktTraj->lGetValue(lTime), psymAggreg
}

long          MARKOVLV::vGetState(long lTime)
{
  if(psymAktTraj != NULL) return(psymAktTraj->lGetValue(lTime));
  else return(-1l);
}

double        MARKOVLV::dGetRandCF(long lTime)
{
  if(psymAktCF != NULL) return(psymAktCF->dGetValue(lTime));
  else return(-1.);
}

double        MARKOVLV::dGetRandDK(long lTime, long lMoment)
{
  if(psymAktDK != NULL) return(psymAktDK->dGetValue(lTime));
  else return(-1.);
}

double        MARKOVLV::dGetMeanCF(long lTime, long lState, long lNrSim)
{
  if(lNrSim == 0) return(0);
  while(lNrTrajSim < lNrSim)
    {
      vGenerateTrajectory();
    }

  return(psymAggregCF->dGetValue(lTime,lState)/lNrTrajSim);
}

double        MARKOVLV::dGetMeanDK(long lTime, long lState, long lNrSim)
{
  if(lNrSim == 0) return(0);
  while(lNrTrajSim < lNrSim)
    {
      vGenerateTrajectory();
    }

  return(psymAggregDK->dGetValue(lTime,lState)/lNrTrajSim);
}

void          MARKOVLV::vNewSeed(long lSeed)
{
  this->lSeed = lSeed;
}

void          MARKOVLV::vResetMeanResults()
```

```
{
  if(psymAktTraj  != NULL) delete(psymAktTraj);
  psymAktTraj = NULL;
  if(psymAktCF    != NULL) delete(psymAktCF);
  psymAktCF = NULL;
  if(psymAktDK    != NULL) delete(psymAktDK);
  psymAktDK = NULL;
  if(psymAggregCF != NULL) delete(psymAggregCF);
  psymAggregCF = NULL;
  if(psymAggregDK != NULL) delete(psymAggregDK);
  psymAggregDK = NULL;
  if(psymAktDisc  != NULL) delete(psymAktDisc);
  psymAktDisc = NULL;
  lNrTrajSim      = 0l;
}

void MARKOVLV::vPrintTeX(FILE * psymTeXFile, bool bWithHeader, char * pcTitle, bool bAllEntries)
{

  char *pcTeXStart[]=
  {
"\\documentclass[11pt,a4paper]{article} \n",
"\\typeout{=====================================================================}\n",
"%%=====================================================================\n",
"%%\\usepackage{cl2emono}\n",
"%%\\usepackage{mkfoot}\n",
"\\usepackage[T1]{fontenc}%%\n",
"\\usepackage[ansinew]{inputenc,}%%\n",
"\\usepackage{a4}%%\n",
"\\usepackage{makeidx,multind,mathcomp}\n",
"\\usepackage{enumerate}%%\n",
"\\usepackage{color}%%\n",
"\\usepackage{amsfonts,amsmath,amssymb,amsthm,txfonts} %%%% txfonts hinzugef\\""{u}gt 2.12.2002\n",
"\\usepackage{supertabular}%%\n",
"\\usepackage[pdftex]{graphicx}\n",
"%%=====================================================================\n",
"\\usepackage[pdftex,\n",
"            colorlinks=true,            %% Schrift von Links in Farbe (true), sonst mit Rahmen (false)
"            bookmarksnumbered=true,     %% Lesezeichen im pdf mit Nummerierung\n",
"            bookmarksopen=true,         %% Öffnet die Lesezeichen vom pdf beim Start\n",
"            bookmarksopenlevel=0,       %% Default ist maxdim\n",
"            pdfstartview=FitH,          %% startet mit Seitenbreite\n",
"            linkcolor=blue,             %% Standard 'red'\n",
"            citecolor=blue,             %% Standard 'green'\n",
"            urlcolor=cyan,              %% Standard 'cyan'\n",
"            filecolor=blue,             %%\n",
"            plainpages=false,pdfpagelabels]{hyperref} %%\n",
"%%%%=====================================================================\n",
"\\parindent 0mm\n",
//        "\\documentclass[a4paper]{article}\n",
    "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    "%% Document generated by OMARKOV.cpp \n",
//   "\\usepackage[latin1]{inputenc}\n",
//    "\\usepackage{a4,latexsym}\n",
    "\\begin{document}\n",
    "\\begin{small} \n",
    ""
  };
  char *pcTeXEnd[]=
  {
    "\\end{small}\n",
    "\\end{document}\n",
    ""
  };
  char *pcTeXTabular[]=
```

```
  {
    "%% ---------- Start Tabular  %s ---------- \n", /* Agument zB Seite 5 */
    "\\par \\bigskip      \n",
    "\\begin{tabular}{%s} \n",  /* Argument z.B. llll */
    "\\end{tabular}       \n",
    "\\par \\bigskip      \n",
    "%% ---------- End Tabular ---------- \n"
  };

  char * pcTeXSections[]=
  {
    "\\section{%s} \n", /* Argument Titel */
    "\\subsection{General} \n",
    "\\newpage \\subsection{Probabilites} \n",
    "\\newpage \\subsection{Technical Interest} \n",
    "\\newpage \\subsection{Prenummerando Benefits} \n",
    "\\newpage \\subsection{Postnummerando Benefits} \n",
    "\\newpage \\subsection{Mathematical Reserves} \n",
    "\\newpage \\subsection{Cash Flows} \n"
  };

#define BOXBREITE  1.5cm
  char pcTableEntry[] = "\\makebox[1.5cm][r]{%s}";
  char pcTableEntryL[] = "\\makebox[1.5cm][r]{%ld}";
  int iC1;
  long lC1, lC2, lTime, lTime2;
  long lState1[MAX_TEX_ROWS];
  long lState2[MAX_TEX_ROWS];
  bool bFillRows;
  long lRowNumber;
  bool bOrgbGetData = bGetData;
  bGetData = true;
  /* 1. Header */
  fprintf(psymTeXFile,strPrgVersionStatic);
  if(bWithHeader)
    {
      for(iC1=0; *pcTeXStart[iC1]; ++ iC1)
        fprintf(psymTeXFile,pcTeXStart[iC1]);
    }

  /* 2. Allgemein */
  fprintf(psymTeXFile,pcTeXSections[0],pcTitle);
  fprintf(psymTeXFile,pcTeXSections[1]);
  fprintf(psymTeXFile,pcTeXTabular[0],"General");
  fprintf(psymTeXFile,pcTeXTabular[1]);
  fprintf(psymTeXFile,pcTeXTabular[2],"ll");
  fprintf(psymTeXFile,"Max. Time  & %ld \\\\ \n",lGetMaxTime());
  fprintf(psymTeXFile,"Nr States  & %ld \\\\ \n",lGetNrStates());
  fprintf(psymTeXFile,"Start Time & %ld \\\\ \n",lGetStartTime());
  fprintf(psymTeXFile,"Stop Time  & %ld \\\\ \n",lGetStopTime());
  fprintf(psymTeXFile,"Prg Version  & %s \\\\ \n",strPrgVersionStatic+1);
  fprintf(psymTeXFile,pcTeXTabular[3]);
  fprintf(psymTeXFile,pcTeXTabular[4]);
  fprintf(psymTeXFile,pcTeXTabular[5]);


  /* 3. Probabilities */
  fprintf(psymTeXFile,pcTeXSections[2]);
  bFillRows  = true;
  lRowNumber = 0;
  for(lC1=0; lC1<lGetNrStates(); ++ lC1)
    {
      for(lC2=0; lC2<lGetNrStates(); ++ lC2)
        {
          if(bFillRows)
```

```
    {
      bool bCheck = false;
      for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
        {
          double dValue = dSetPij(lTime, lC1, lC2, 0.);
          if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
        }

      if(bAllEntries) bCheck = true;
      if(bCheck){
        lState1[lRowNumber] =lC1;
        lState2[lRowNumber] =lC2;
        ++lRowNumber;
        if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
      }
    }
  else
    {
      lTime2 = 0;
      fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
      fprintf(psymTeXFile,pcTeXTabular[1]);
      fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
      fprintf(psymTeXFile,"From ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      fprintf(psymTeXFile,"To ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
        {
          fprintf(psymTeXFile,"%d ", lTime);
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            vPrintTexNumber(psymTeXFile, dSetPij(lTime, lState1[iC1], lState2[iC1], 0.));

          fprintf(psymTeXFile,"\\\\ \n");
          if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
            {fprintf(psymTeXFile,pcTeXTabular[3]);
            fprintf(psymTeXFile,pcTeXTabular[4]);
            fprintf(psymTeXFile,pcTeXTabular[5]);
            fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
            fprintf(psymTeXFile,pcTeXTabular[1]);
            fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
            fprintf(psymTeXFile,"From ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            fprintf(psymTeXFile,"To ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            lTime2 = -1;
```

```
                    }
                  ++lTime2;
                }
              fprintf(psymTeXFile,pcTeXTabular[3]);
              fprintf(psymTeXFile,pcTeXTabular[4]);
              fprintf(psymTeXFile,pcTeXTabular[5]);
              bFillRows  = true;
              lRowNumber = 0;
              if(bFillRows)
                {
                  bool bCheck = false;
                  for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                    {
                      double dValue = dSetPij(lTime, lC1, lC2, 0.);
                      if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
                    }

                  if(bAllEntries) bCheck = true;
                  if(bCheck){
                    lState1[lRowNumber] =lC1;
                    lState2[lRowNumber] =lC2;
                    ++lRowNumber;
                    if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
                  }
                }

            }
        }
    }
  if(lRowNumber > 0)
    {
      lTime2 = 0;
      fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
      fprintf(psymTeXFile,pcTeXTabular[1]);
      fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
      fprintf(psymTeXFile,"From ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      fprintf(psymTeXFile,"To ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
        {
          fprintf(psymTeXFile,"%d ", lTime);
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
             vPrintTexNumber(psymTeXFile, dSetPij(lTime, lState1[iC1], lState2[iC1], 0.));
      fprintf(psymTeXFile,"\\\\ \n");
          if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
            {fprintf(psymTeXFile,pcTeXTabular[3]);
            fprintf(psymTeXFile,pcTeXTabular[4]);
            fprintf(psymTeXFile,pcTeXTabular[5]);
            fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
            fprintf(psymTeXFile,pcTeXTabular[1]);
            fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
            fprintf(psymTeXFile,"From ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
```

```
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            fprintf(psymTeXFile,"To ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            lTime2 = -1;
            }
          ++lTime2;
        }
      fprintf(psymTeXFile,pcTeXTabular[3]);
      fprintf(psymTeXFile,pcTeXTabular[4]);
      fprintf(psymTeXFile,pcTeXTabular[5]);
      bFillRows  = true;
      lRowNumber = 0;
    }

  /* 3. Disconts  */
  fprintf(psymTeXFile,pcTeXSections[3]);
  bFillRows  = true;
  lRowNumber = 0;
  for(lC1=0; lC1<lGetNrStates(); ++ lC1)
    {
      long lStartLoc, lStopLoc;
      if(bStochasticInterest)
        {
          lStartLoc = 0;
          lStopLoc  = lGetNrStates();
        }
      else
        {
          lStartLoc = lC1;
          lStopLoc  = lStartLoc + 1;
        }
      for(lC2=lStartLoc; lC2<lStopLoc; ++ lC2)
        {
          if(bFillRows)
            {
              bool bCheck = false;
              for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                {
                  double dValue = dSetDisc(lTime, lC1, lC2, 0.);
                  if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
                }

              if(bAllEntries) bCheck = true;
              if(bCheck){
                lState1[lRowNumber] =lC1;
                lState2[lRowNumber] =lC2;
                ++lRowNumber;
                if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
              }
            }
          else
            {
              lTime2 = 0;
              fprintf(psymTeXFile,pcTeXTabular[0],"Disc.");
              fprintf(psymTeXFile,pcTeXTabular[1]);
              fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
              fprintf(psymTeXFile,"From ");
```

```
  for(iC1=0; iC1 < lRowNumber; ++ iC1)
    {
      fprintf(psymTeXFile," & ");
      fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
    }
  fprintf(psymTeXFile,"\\\\ \n");
  fprintf(psymTeXFile,"To ");
  for(iC1=0; iC1 < lRowNumber; ++ iC1)
    {
      fprintf(psymTeXFile," & ");
      fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
    }
  fprintf(psymTeXFile,"\\\\ \n");
  for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
    {
      fprintf(psymTeXFile,"%d ", lTime);
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
vPrintTexNumber(psymTeXFile, dSetDisc(lTime, lState1[iC1], lState2[iC1], 0.));
      fprintf(psymTeXFile,"\\\\ \n");
      if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
        {fprintf(psymTeXFile,pcTeXTabular[3]);
        fprintf(psymTeXFile,pcTeXTabular[4]);
        fprintf(psymTeXFile,pcTeXTabular[5]);
        fprintf(psymTeXFile,pcTeXTabular[0],"Disc.");
        fprintf(psymTeXFile,pcTeXTabular[1]);
        fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
        fprintf(psymTeXFile,"From ");
        for(iC1=0; iC1 < lRowNumber; ++ iC1)
          {
            fprintf(psymTeXFile," & ");
            fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
          }
        fprintf(psymTeXFile,"\\\\ \n");
        fprintf(psymTeXFile,"To ");
        for(iC1=0; iC1 < lRowNumber; ++ iC1)
          {
            fprintf(psymTeXFile," & ");
            fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
          }
        fprintf(psymTeXFile,"\\\\ \n");
        lTime2 = -1;
        }
      ++lTime2;
    }
  fprintf(psymTeXFile,pcTeXTabular[3]);
  fprintf(psymTeXFile,pcTeXTabular[4]);
  fprintf(psymTeXFile,pcTeXTabular[5]);
  bFillRows  = true;
  lRowNumber = 0;
  if(bFillRows)
    {
      bool bCheck = false;
      for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
        {
          double dValue = dSetDisc(lTime, lC1, lC2, 0.);
          if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
        }

      if(bAllEntries) bCheck = true;
      if(bCheck){
        lState1[lRowNumber] =lC1;
        lState2[lRowNumber] =lC2;
        ++lRowNumber;
        if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
      }
```

```
                }
            }
        }
    }
  if(lRowNumber > 0)
    {
      lTime2 = 0;
      fprintf(psymTeXFile,pcTeXTabular[0],"Disc.");
      fprintf(psymTeXFile,pcTeXTabular[1]);
      fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
      fprintf(psymTeXFile,"From ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      fprintf(psymTeXFile,"To ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
        {
          fprintf(psymTeXFile,"%d ", lTime);
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            vPrintTexNumber(psymTeXFile, dSetDisc(lTime, lState1[iC1], lState2[iC1], 0.));
          fprintf(psymTeXFile,"\\\\ \n");
          if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
            {fprintf(psymTeXFile,pcTeXTabular[3]);
            fprintf(psymTeXFile,pcTeXTabular[4]);
            fprintf(psymTeXFile,pcTeXTabular[5]);
            fprintf(psymTeXFile,pcTeXTabular[0],"Disc.");
            fprintf(psymTeXFile,pcTeXTabular[1]);
            fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
            fprintf(psymTeXFile,"From ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            fprintf(psymTeXFile,"To ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            lTime2 = -1;
            }
          ++lTime2;
        }
      fprintf(psymTeXFile,pcTeXTabular[3]);
      fprintf(psymTeXFile,pcTeXTabular[4]);
      fprintf(psymTeXFile,pcTeXTabular[5]);
      bFillRows  = true;
      lRowNumber = 0;
    }

  /* 4. Prenummerando */
  fprintf(psymTeXFile,pcTeXSections[4]);
  bFillRows  = true;
```

```
  lRowNumber = 0;
  for(lC1=0; lC1<lGetNrStates(); ++ lC1)
    {
      //        for(lC2=0; lC2<lGetNrStates(); ++ lC2)
      //        {
      if(bFillRows)
        {
          bool bCheck = false;
          for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
            {
              double dValue = dSetPre(lTime, lC1, lC1, 0.);
              if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
            }

          if(bAllEntries) bCheck = true;
          if(bCheck){
            lState1[lRowNumber] =lC1;
            lState2[lRowNumber] =lC1;
            ++lRowNumber;
            if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
          }
        }
      else
        {
          lTime2 = 0;
          fprintf(psymTeXFile,pcTeXTabular[0],"Pre");
          fprintf(psymTeXFile,pcTeXTabular[1]);
          fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
          fprintf(psymTeXFile,"From ");
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            {
              fprintf(psymTeXFile," & ");
              fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
            }
          fprintf(psymTeXFile,"\\\\ \n");
          fprintf(psymTeXFile,"To ");
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            {
              fprintf(psymTeXFile," & ");
              fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
            }
          fprintf(psymTeXFile,"\\\\ \n");
          for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
            {
              fprintf(psymTeXFile,"%d ", lTime);
              for(iC1=0; iC1 < lRowNumber; ++ iC1)
                     vPrintTexNumber(psymTeXFile, dSetPre(lTime, lState1[iC1], lState2[iC1], 0.));

              fprintf(psymTeXFile,"\\\\ \n");
              if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
                {fprintf(psymTeXFile,pcTeXTabular[3]);
                fprintf(psymTeXFile,pcTeXTabular[4]);
                fprintf(psymTeXFile,pcTeXTabular[5]);
                fprintf(psymTeXFile,pcTeXTabular[0],"Pre");
                fprintf(psymTeXFile,pcTeXTabular[1]);
                fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
                fprintf(psymTeXFile,"From ");
                for(iC1=0; iC1 < lRowNumber; ++ iC1)
                  {
                    fprintf(psymTeXFile," & ");
                    fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
                  }
                fprintf(psymTeXFile,"\\\\ \n");
                fprintf(psymTeXFile,"To ");
                for(iC1=0; iC1 < lRowNumber; ++ iC1)
```

```
                {
                  fprintf(psymTeXFile," & ");
                  fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
                }
              fprintf(psymTeXFile,"\\\\ \n");
              lTime2 = -1;
              }
            ++lTime2;
          }
        fprintf(psymTeXFile,pcTeXTabular[3]);
        fprintf(psymTeXFile,pcTeXTabular[4]);
        fprintf(psymTeXFile,pcTeXTabular[5]);
        bFillRows  = true;
        lRowNumber = 0;
        if(bFillRows)
          {
            bool bCheck = false;
            for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
              {
                double dValue = dSetPre(lTime, lC1, lC1, 0.);
                if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
              }

            if(bAllEntries) bCheck = true;
            if(bCheck){
              lState1[lRowNumber] =lC1;
              lState2[lRowNumber] =lC1;
              ++lRowNumber;
              if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
            }
          }
      }
    //        }
  }
if(lRowNumber > 0)
  {
    lTime2 = 0;
    fprintf(psymTeXFile,pcTeXTabular[0],"Pre");
    fprintf(psymTeXFile,pcTeXTabular[1]);
    fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
    fprintf(psymTeXFile,"From ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    fprintf(psymTeXFile,"To ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
      {
        fprintf(psymTeXFile,"%d ", lTime);
        for(iC1=0; iC1 < lRowNumber; ++ iC1)
        vPrintTexNumber(psymTeXFile, dSetPre(lTime, lState1[iC1], lState2[iC1], 0.));
    fprintf(psymTeXFile,"\\\\ \n");
        if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
          {fprintf(psymTeXFile,pcTeXTabular[3]);
          fprintf(psymTeXFile,pcTeXTabular[4]);
          fprintf(psymTeXFile,pcTeXTabular[5]);
          fprintf(psymTeXFile,pcTeXTabular[0],"Pre");
```

```
            fprintf(psymTeXFile,pcTeXTabular[1]);
            fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
            fprintf(psymTeXFile,"From ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            fprintf(psymTeXFile,"To ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            lTime2 = -1;
            }
          ++lTime2;
        }
      fprintf(psymTeXFile,pcTeXTabular[3]);
      fprintf(psymTeXFile,pcTeXTabular[4]);
      fprintf(psymTeXFile,pcTeXTabular[5]);
      bFillRows  = true;
      lRowNumber = 0;
    }

  /* 5. Postnummerando */
  fprintf(psymTeXFile,pcTeXSections[5]);
  bFillRows  = true;
  lRowNumber = 0;
  for(lC1=0; lC1<lGetNrStates(); ++ lC1)
    {
      for(lC2=0; lC2<lGetNrStates(); ++ lC2)
        {
          if(bFillRows)
            {
              bool bCheck = false;
              for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                {
                  double dValue = dSetPost(lTime, lC1, lC2, 0.);
                  if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
                }

              if(bAllEntries) bCheck = true;
              if(bCheck){
                lState1[lRowNumber] =lC1;
                lState2[lRowNumber] =lC2;
                ++lRowNumber;
                if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
              }
            }
          else
            {
              lTime2 = 0;
              fprintf(psymTeXFile,pcTeXTabular[0],"Post");
              fprintf(psymTeXFile,pcTeXTabular[1]);
              fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
              fprintf(psymTeXFile,"From ");
              for(iC1=0; iC1 < lRowNumber; ++ iC1)
                {
                  fprintf(psymTeXFile," & ");
                  fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
                }
              fprintf(psymTeXFile,"\\\\ \n");
```

```
                fprintf(psymTeXFile,"To ");
                for(iC1=0; iC1 < lRowNumber; ++ iC1)
                  {
                    fprintf(psymTeXFile," & ");
                    fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
                  }
                fprintf(psymTeXFile,"\\\\ \n");
                for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                  {
                    fprintf(psymTeXFile,"%d ", lTime);
                    for(iC1=0; iC1 < lRowNumber; ++ iC1)
                   vPrintTexNumber(psymTeXFile, dSetPost(lTime, lState1[iC1], lState2[iC1], 0.));
                      fprintf(psymTeXFile,"\\\\ \n");
                    if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
                      {fprintf(psymTeXFile,pcTeXTabular[3]);
                      fprintf(psymTeXFile,pcTeXTabular[4]);
                      fprintf(psymTeXFile,pcTeXTabular[5]);
                      fprintf(psymTeXFile,pcTeXTabular[0],"Post");
                      fprintf(psymTeXFile,pcTeXTabular[1]);
                      fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
                      fprintf(psymTeXFile,"From ");
                      for(iC1=0; iC1 < lRowNumber; ++ iC1)
                        {
                          fprintf(psymTeXFile," & ");
                          fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
                        }
                      fprintf(psymTeXFile,"\\\\ \n");
                      fprintf(psymTeXFile,"To ");
                      for(iC1=0; iC1 < lRowNumber; ++ iC1)
                        {
                          fprintf(psymTeXFile," & ");
                          fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
                        }
                      fprintf(psymTeXFile,"\\\\ \n");
                      lTime2 = -1;
                      }
                    ++lTime2;
                  }
                fprintf(psymTeXFile,pcTeXTabular[3]);
                fprintf(psymTeXFile,pcTeXTabular[4]);
                fprintf(psymTeXFile,pcTeXTabular[5]);
                bFillRows  = true;
                lRowNumber = 0;
                if(bFillRows)
                  {
                    bool bCheck = false;
                    for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                      {
                        double dValue = dSetPost(lTime, lC1, lC2, 0.);
                        if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
                      }

                    if(bAllEntries) bCheck = true;
                    if(bCheck){
                      lState1[lRowNumber] =lC1;
                      lState2[lRowNumber] =lC2;
                      ++lRowNumber;
                      if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
                    }
                  }
              }

        }
    }
  if(lRowNumber > 0)
```

```
  {
    lTime2 = 0;
    fprintf(psymTeXFile,pcTeXTabular[0],"Post");
    fprintf(psymTeXFile,pcTeXTabular[1]);
    fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
    fprintf(psymTeXFile,"From ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    fprintf(psymTeXFile,"To ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
      {
        fprintf(psymTeXFile,"%d ", lTime);
        for(iC1=0; iC1 < lRowNumber; ++ iC1)
        vPrintTexNumber(psymTeXFile, dSetPost(lTime, lState1[iC1], lState2[iC1], 0.));

        fprintf(psymTeXFile,"\\\\ \n");
        if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
          {fprintf(psymTeXFile,pcTeXTabular[3]);
          fprintf(psymTeXFile,pcTeXTabular[4]);
          fprintf(psymTeXFile,pcTeXTabular[5]);
          fprintf(psymTeXFile,pcTeXTabular[0],"Post");
          fprintf(psymTeXFile,pcTeXTabular[1]);
          fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
          fprintf(psymTeXFile,"From ");
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            {
              fprintf(psymTeXFile," & ");
              fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
            }
          fprintf(psymTeXFile,"\\\\ \n");
          fprintf(psymTeXFile,"To ");
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            {
              fprintf(psymTeXFile," & ");
              fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
            }
          fprintf(psymTeXFile,"\\\\ \n");
          lTime2 = -1;
          }
        ++lTime2;
      }
    fprintf(psymTeXFile,pcTeXTabular[3]);
    fprintf(psymTeXFile,pcTeXTabular[4]);
    fprintf(psymTeXFile,pcTeXTabular[5]);
    bFillRows  = true;
    lRowNumber = 0;
  }

/* 6. MR */
fprintf(psymTeXFile,pcTeXSections[6]);
bFillRows  = true;
lRowNumber = 0;
for(lC1=1; lC1<=lDKCalculated; ++ lC1)
  {
    for(lC2=0; lC2<lGetNrStates(); ++ lC2)
```

```
{
  if(bFillRows)
    {
      lState1[lRowNumber] =lC1;
      lState2[lRowNumber] =lC2;
      ++lRowNumber;
      if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
    }
  else
    {
      lTime2 = 0;
      fprintf(psymTeXFile,pcTeXTabular[0],"MR.");
      fprintf(psymTeXFile,pcTeXTabular[1]);
      fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
      fprintf(psymTeXFile,"Moment ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      fprintf(psymTeXFile,"State ");
      for(iC1=0; iC1 < lRowNumber; ++ iC1)
        {
          fprintf(psymTeXFile," & ");
          fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
        }
      fprintf(psymTeXFile,"\\\\ \n");
      for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
        {
          fprintf(psymTeXFile,"%d ", lTime);
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
          vPrintTexNumber(psymTeXFile, dGetDK(lTime, lState2[iC1], lState1[iC1]));

          fprintf(psymTeXFile,"\\\\ \n");
          if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
            {fprintf(psymTeXFile,pcTeXTabular[3]);
            fprintf(psymTeXFile,pcTeXTabular[4]);
            fprintf(psymTeXFile,pcTeXTabular[5]);
            fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
            fprintf(psymTeXFile,pcTeXTabular[1]);
            fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
            fprintf(psymTeXFile,"From ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            fprintf(psymTeXFile,"To ");
            for(iC1=0; iC1 < lRowNumber; ++ iC1)
              {
                fprintf(psymTeXFile," & ");
                fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
              }
            fprintf(psymTeXFile,"\\\\ \n");
            lTime2 = -1;
            }
          ++lTime2;
        }
      fprintf(psymTeXFile,pcTeXTabular[3]);
      fprintf(psymTeXFile,pcTeXTabular[4]);
      fprintf(psymTeXFile,pcTeXTabular[5]);
      bFillRows  = true;
      lRowNumber = 0;
```

```
            if(bFillRows)
              {
                lState1[lRowNumber] =lC1;
                lState2[lRowNumber] =lC2;
                ++lRowNumber;
                if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
              }
          }

      }
  }
if(lRowNumber > 0)
  {
    lTime2 = 0;
    fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
    fprintf(psymTeXFile,pcTeXTabular[1]);
    fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
    fprintf(psymTeXFile,"Moment ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    fprintf(psymTeXFile,"State ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
      {
        fprintf(psymTeXFile,"%d ", lTime);
        for(iC1=0; iC1 < lRowNumber; ++ iC1)
                vPrintTexNumber(psymTeXFile, dGetDK(lTime, lState2[iC1], lState1[iC1]));
        fprintf(psymTeXFile,"\\\\ \n");
        if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
          {fprintf(psymTeXFile,pcTeXTabular[3]);
          fprintf(psymTeXFile,pcTeXTabular[4]);
          fprintf(psymTeXFile,pcTeXTabular[5]);
          fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
          fprintf(psymTeXFile,pcTeXTabular[1]);
          fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
          fprintf(psymTeXFile,"From ");
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            {
              fprintf(psymTeXFile," & ");
              fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
            }
          fprintf(psymTeXFile,"\\\\ \n");
          fprintf(psymTeXFile,"To ");
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
            {
              fprintf(psymTeXFile," & ");
              fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
            }
          fprintf(psymTeXFile,"\\\\ \n");
          lTime2 = -1;
          }
        ++lTime2;
      }
    fprintf(psymTeXFile,pcTeXTabular[3]);
    fprintf(psymTeXFile,pcTeXTabular[4]);
    fprintf(psymTeXFile,pcTeXTabular[5]);
```

```
      bFillRows  = true;
      lRowNumber = 0;
    }

  /* 7. CF */
  fprintf(psymTeXFile,pcTeXSections[7]);
  bFillRows  = true;
  lRowNumber = 0;
  for(lC1=0; lC1<lCFCalculated; ++ lC1)
    {
      for(lC2=0; lC2<lGetNrStates(); ++ lC2)
        {
          if(bFillRows)
            {
              bool bCheck = false;
              for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                {
                  double dValue = dGetCF(lTime, lC1, lC2);
                  if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
                }

              if(bAllEntries) bCheck = true;
              if(bCheck){
                lState1[lRowNumber] =lC1;
                lState2[lRowNumber] =lC2;
                ++lRowNumber;
                if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
              }
            }
          else
            {
              lTime2 = 0;
              fprintf(psymTeXFile,pcTeXTabular[0],"From");
              fprintf(psymTeXFile,pcTeXTabular[1]);
              fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
              fprintf(psymTeXFile,"Moment ");
              for(iC1=0; iC1 < lRowNumber; ++ iC1)
                {
                  fprintf(psymTeXFile," & ");
                  fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
                }
              fprintf(psymTeXFile,"\\\\ \n");
              fprintf(psymTeXFile,"To ");
              for(iC1=0; iC1 < lRowNumber; ++ iC1)
                {
                  fprintf(psymTeXFile," & ");
                  fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
                }
              fprintf(psymTeXFile,"\\\\ \n");
              for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                {
                  fprintf(psymTeXFile,"%d ", lTime);
                  for(iC1=0; iC1 < lRowNumber; ++ iC1)
                  vPrintTexNumber(psymTeXFile, dGetCF(lTime, lState1[iC1], lState2[iC1]));

                  fprintf(psymTeXFile,"\\\\ \n");
                  if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
                    {fprintf(psymTeXFile,pcTeXTabular[3]);
                    fprintf(psymTeXFile,pcTeXTabular[4]);
                    fprintf(psymTeXFile,pcTeXTabular[5]);
                    fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
                    fprintf(psymTeXFile,pcTeXTabular[1]);
                    fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
                    fprintf(psymTeXFile,"From ");
                    for(iC1=0; iC1 < lRowNumber; ++ iC1)
```

```
                    {
                      fprintf(psymTeXFile," & ");
                      fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
                    }
                  fprintf(psymTeXFile,"\\\\ \n");
                  fprintf(psymTeXFile,"To ");
                  for(iC1=0; iC1 < lRowNumber; ++ iC1)
                    {
                      fprintf(psymTeXFile," & ");
                      fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
                    }
                  fprintf(psymTeXFile,"\\\\ \n");
                  lTime2 = -1;
                  }
                ++lTime2;
              }
            fprintf(psymTeXFile,pcTeXTabular[3]);
            fprintf(psymTeXFile,pcTeXTabular[4]);
            fprintf(psymTeXFile,pcTeXTabular[5]);
            bFillRows  = true;
            lRowNumber = 0;

            if(bFillRows)
              {
                bool bCheck = false;
                for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
                  {
                    double dValue = dGetCF(lTime, lC1, lC2);
                    if (fabs(dValue) > TECHEPS) {bCheck = true; break;}
                  }

                if(bAllEntries) bCheck = true;
                if(bCheck){
                  lState1[lRowNumber] =lC1;
                  lState2[lRowNumber] =lC2;
                  ++lRowNumber;
                  if(lRowNumber >= MAX_TEX_ROWS) bFillRows  = false;
                }
              }
          }
      }
  }
if(lRowNumber > 0)
  {
    lTime2 = 0;
    fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
    fprintf(psymTeXFile,pcTeXTabular[1]);
    fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
    fprintf(psymTeXFile,"From ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    fprintf(psymTeXFile,"To ");
    for(iC1=0; iC1 < lRowNumber; ++ iC1)
      {
        fprintf(psymTeXFile," & ");
        fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
      }
    fprintf(psymTeXFile,"\\\\ \n");
    for (lTime=lGetStopTime(); lTime <= lGetStartTime(); ++lTime)
      {
        fprintf(psymTeXFile,"%d ", lTime);
```

```
          for(iC1=0; iC1 < lRowNumber; ++ iC1)
                  vPrintTexNumber(psymTeXFile, dGetCF(lTime, lState1[iC1], lState2[iC1]));
          fprintf(psymTeXFile,"\\\\ \n");
          if(lTime2>=MAX_TEX_LINES && lTime !=lGetStartTime())
            {
              fprintf(psymTeXFile,pcTeXTabular[3]);
              fprintf(psymTeXFile,pcTeXTabular[4]);
              fprintf(psymTeXFile,pcTeXTabular[5]);
              fprintf(psymTeXFile,pcTeXTabular[0],"Prob.");
              fprintf(psymTeXFile,pcTeXTabular[1]);
              fprintf(psymTeXFile,pcTeXTabular[2],TEX_TAB);
              fprintf(psymTeXFile,"From ");
              for(iC1=0; iC1 < lRowNumber; ++ iC1)
                {
                  fprintf(psymTeXFile," & ");
                  fprintf(psymTeXFile,pcTableEntryL,lState1[iC1]);
                }
              fprintf(psymTeXFile,"\\\\ \n");
              fprintf(psymTeXFile,"To ");
              for(iC1=0; iC1 < lRowNumber; ++ iC1)
                {
                  fprintf(psymTeXFile," & ");
                  fprintf(psymTeXFile,pcTableEntryL,lState2[iC1]);
                }
              fprintf(psymTeXFile,"\\\\ \n");
              lTime2 = -1;
            }
          ++lTime2;
        }
      fprintf(psymTeXFile,pcTeXTabular[3]);
      fprintf(psymTeXFile,pcTeXTabular[4]);
      fprintf(psymTeXFile,pcTeXTabular[5]);
      bFillRows  = true;
      lRowNumber = 0;
    }
  /* 8. Footer */
  if(bWithHeader)
    {
      for(iC1=0; *pcTeXEnd[iC1]; ++ iC1)
        fprintf(psymTeXFile,pcTeXEnd[iC1]);
    }
  bGetData = bOrgbGetData;
}
```

Chapter 5
annuity.h

```
////////////////////////////////////////////////////////////////////
//                                                                //
// Annuity LV Zahlungsstroeme                                     //
//                                                                //
// Autor Michael Koller                                           //
//                                                                //
// Datum 3.2011: erstellt                                         //
//                                                                //
////////////////////////////////////////////////////////////////////

#ifndef  _OANNUITY_INCLUDED
#define _OANNUITY_INCLUDED

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include "omarkov.h"


 class ANNUITYLV:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  ANNUITYLV();
  ANNUITYLV(long lMaxTimesIpt, double dPre); // Overrides Defaults
  ~ANNUITYLV();
  int             iSetTable(char * pcId);
  void            vSetStartTime(long lTime);
  void            vSetStopTime(long lTime);
  void            vSetSAge(long lTime);
  void            vSetG(long lTime);
  void            vSetMaxProj(long lMaxYearImp);
  double          dSetQx(long lTime, double dValue);  // lNach irrelevant
  double          dSetFx(long lTime, double dValue);
  double          dSetSx(long lTime, double dValue);
  double          dSetBaseYear(long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);
  double          dGetDK(long lTime); // Berechnet DK's
  double          dGetCF(long lTime);
  double          dGetQx(long lTime, long lYear);
  double          dGetTqx(long lTime);
  double          dGetTpx(long lTime);
  double          dSetPre(double dValue);
  double          dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx
  void            vLeistReset();
```

```
  void            vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncrement);
  void            vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor); // eac
private:
  long            lGTime;
  long            lSAge;
  long            lMaxProj;
  LV_VECTOR       *psymQx;
  LV_VECTOR       *psymFx;
  LV_VECTOR       *psymSx;
  LV_VECTOR       *psymDisc;
  LV_VECTOR       *psymRelQxTime;
  LV_VECTOR       *psymBenefit;
  long            lBaseYear;
  long            lActualYear;
  bool            lValid;
  long            lMaxTime;
  double          dPre;
  double          dPost;
};

#endif
```

Chapter 6

annuity.cpp

```
////////////////////////////////////////////////////////////////////
//                                                                //
// Annuity LV Zahlungsstroeme                                     //
//                                                                //
// Autor Michael Koller                                           //
//                                                                //
// Datum 3.2011: erstellt                                         //
//                                                                //
////////////////////////////////////////////////////////////////////

#include "annuity.h"
#ifdef FOR_OLE
#pragma message ("In order to avoid double references to omarkov.cpp we include only header")
#include "omarkov.h"
#else
#include "omarkov.cpp"
#endif

ANNUITYLV::ANNUITYLV():MARKOVLV(250l,3l,1l)
{
  int iC1;
  double dTemp;
  //  MARKOVLV::MARKOVLV(250l,3l,1l);
  vSetNrStates(3l);
  lGTime =0l;
  lSAge  =0l;
  lMaxProj = 10000l;
  psymQx = new LV_VECTOR(250, 0, 0);
  psymFx = new LV_VECTOR(250, 0, 0);
  psymSx = new LV_VECTOR(250, 0, 0);
  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymRelQxTime = new LV_VECTOR(250, 0, 0);
  psymBenefit  = new LV_VECTOR(250, 0, 0);
  lBaseYear = 2000l;
  lActualYear = 2000l;
  lValid = false;
  lMaxTime = 250;
  this->dPre = 1.;
  dPost = 1. - dPre;
  for (iC1 = 0; iC1 < 250; ++iC1)
    {
      dTemp = psymRelQxTime->dSetValue(iC1, 1.);
      dTemp = psymBenefit->dSetValue(iC1, 1.);
    }
}
```

```
ANNUITYLV::ANNUITYLV(long lMaxTimesIpt, double dPre):MARKOVLV(250l,3l,1l)
{
  int iC1;
  double dTemp;
  //  MARKOVLV::MARKOVLV(250l,3l,1l);
  vSetNrStates(3l);
  lGTime =0l;
  lSAge  =0l;
  lMaxProj = 10000l;
  psymQx = new LV_VECTOR(250, 0, 0);
  psymFx = new LV_VECTOR(250, 0, 0);
  psymSx = new LV_VECTOR(250, 0, 0);
  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymRelQxTime = new LV_VECTOR(250, 0, 0);
  psymBenefit  = new LV_VECTOR(250, 0, 0);
  lBaseYear = 2000l;
  lActualYear = 2000l;
  lValid = false;
  lMaxTime = lMaxTimesIpt;
  this->dPre = dPre;
  dPost = 1. - dPre;
  for (iC1 = 0; iC1 < 250; ++iC1)
    {
      dTemp = psymRelQxTime->dSetValue(iC1, 1.);
      dTemp = psymBenefit->dSetValue(iC1, 1.);
    }
}

ANNUITYLV::~ANNUITYLV()
{
  delete(psymQx);
  delete(psymFx);
  delete(psymSx);
  delete(psymDisc);
  delete(psymRelQxTime);
  delete(psymBenefit);
}

int  ANNUITYLV::iSetTable(char * pcId)
{
  int iC1,iC2;
  int iX0;
  int iXOmega;
  double dTech;
  lValid = false;
  psymQx->vReset();
  if (!this->psymTable1)
    this->psymTable1 = new TABLESERVER();
  this->psymTable1->vSetTable(pcId);
  iC2= this->psymTable1->iTableNumber();
  if (iC2<0)
    return(iC2);
  iX0 = this->psymTable1->iX0();
  iXOmega = this->psymTable1->iXOmega();
  vSetStartTime(iXOmega+1);
  dTech = this->psymTable1->dITech();
  for(iC1=0; iC1<2500; ++iC1)
    {
      dSetDisc(iC1, 1./(1.000000000001+ dTech));
    }
  for(iC1=0; iC1 <= iXOmega; ++iC1)
    {
      dSetQx(iC1, this->psymTable1->dGetValue(iC1));
    }
  return(iC2);
```

```
}

void ANNUITYLV::vSetStartTime(long lTime)
{
  MARKOVLV::vSetStartTime(lTime);
}

void ANNUITYLV::vSetStopTime(long lTime)
{
  MARKOVLV::vSetStopTime(lTime);
}

void ANNUITYLV::vSetSAge(long lTime)
{
  lValid = false;
  lSAge  =lTime;
}

void ANNUITYLV::vSetG(long lTime)
{
  lValid = false;
  lGTime = lTime;
}

double ANNUITYLV::dSetQx(long lTime, double dValue)
{
  lValid = false;
  return(psymQx->dSetValue(lTime, dValue));
}

double ANNUITYLV::dSetFx(long lTime, double dValue)
{
  lValid = false;
  return(psymFx->dSetValue(lTime, dValue));
}

double ANNUITYLV::dSetSx(long lTime, double dValue)
{
  lValid = false;
  return(psymSx->dSetValue(lTime, dValue));
}

double ANNUITYLV::dSetBaseYear(long lTime)
{
  lValid = false;
  lBaseYear = lTime;
  return((double) lTime);
}

void ANNUITYLV::vSetMaxProj(long lMaxYearImp)
{
  lValid = false;
  lMaxProj= lMaxYearImp;

}

double ANNUITYLV::dSetActualYear(long lTime)
{
  lValid = false;
  lActualYear = lTime;
  return((double) lTime);
}

double ANNUITYLV::dSetDisc(long lTime, double dValue)
{
```

```
  lValid = false;
  return(psymDisc->dSetValue(lTime, dValue));

}

double ANNUITYLV::dGetDK(long lTime)
{
  int iC1, iC2= MARKOVLV::lGetStopTime();
  double dQxLoc, dTemp, dLocDisc, dLeist;
  if (lValid == true) return(MARKOVLV::dGetDK(lTime,0l,1l));
  // Set Different Markov Elements before Calc
  lValid = true;
  MARKOVLV::vReset();
  // 1. Set Probabilities
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dTemp = psymRelQxTime->dGetValue(iC1 - MARKOVLV::lGetStopTime());
      if(iC1 - MARKOVLV::lGetStopTime() < 0) dTemp  = 1;
      dQxLoc = dTemp  * dGetQx(iC1 , iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      if(dQxLoc < 0) dQxLoc = 0.;
      if(dQxLoc > 1) dQxLoc = 1.;
      dTemp = MARKOVLV::dSetPij(iC1, 0, 0, 1. - dQxLoc);
      dTemp = MARKOVLV::dSetPij(iC1, 1, 1, 1.);
      if(iC1 < lSAge ||  iC1 >= lSAge + lGTime)
        dTemp = MARKOVLV::dSetPij(iC1, 0, 2, dQxLoc); // 2 Zustand ohne Garantie
      else
        dTemp = MARKOVLV::dSetPij(iC1, 0, 1, dQxLoc); // Zustand 1 mit Garantie
    }
  // 2. Set LV
  for(iC1=lSAge; iC1<lMaxTime; ++iC1)
    {
      dLeist = psymBenefit->dGetValue(iC1);
      dTemp = MARKOVLV::dSetPre(iC1, 0, 0, dLeist * dPre);
      dTemp = MARKOVLV::dSetPost(iC1, 0, 0, dLeist * dPost);
    }
  for(iC1=lSAge; iC1<lSAge + lGTime; ++iC1)
    {
      dLeist = psymBenefit->dGetValue(iC1);
      dTemp = MARKOVLV::dSetPre(iC1, 1, 1, dLeist * dPre);
      dTemp = MARKOVLV::dSetPost(iC1, 1, 1, dLeist * dPost);
      dTemp = MARKOVLV::dSetPost(iC1, 0, 1, dLeist * dPost);
    }
  // 3. Set Discount
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dLocDisc = psymDisc->dGetValue(iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      dTemp = MARKOVLV::dSetDisc(iC1, 0, 0, dLocDisc);
      dTemp = MARKOVLV::dSetDisc(iC1, 1, 1, dLocDisc);
      dTemp = MARKOVLV::dSetDisc(iC1, 2, 2, dLocDisc);
    }
  // 4. Calc DK
  return(MARKOVLV::dGetDK(lTime,0l,1l));
}

double ANNUITYLV::dGetCF(long lTime)
{
  double dTemp;
  if (lValid == false) dTemp = ANNUITYLV::dGetDK(lTime);
  return(MARKOVLV::dGetCF(lTime, 0, 0) + MARKOVLV::dGetCF(lTime, 0, 1) );
}

double ANNUITYLV::dGetQx(long lTime, long lYear)
{
  long lDeltaT = lYear - lBaseYear;
  if (lDeltaT > lMaxProj) lDeltaT = lMaxProj;
```

```
  return(psymQx->dGetValue(lTime) * exp(psymFx->dGetValue(lTime) * (lDeltaT)));
}

double  ANNUITYLV::dGetTpx(long lTime)
{
  double dTemp = 1.;
  int iC;
  for(iC=0;iC<lTime; ++iC)
    {
      //printf("\n tpx time %ld  iC %d  dTemp %10.4f  sx %10.4f  qx %10.4f \n", lTime, iC, dTemp, psymS
dGetQx(MARKOVLV::lGetStopTime() +iC,  lActualYear + iC));
      dTemp *= (1.- psymSx->dGetValue(MARKOVLV::lGetStopTime() +iC)) * (1- dGetQx(MARKOVLV::lGetStopTim
lActualYear + iC));
    }
       return(dTemp);
}
double  ANNUITYLV::dGetTqx(long lTime)
{
  return(dGetTpx(lTime) * dGetQx(MARKOVLV::lGetStopTime() +lTime,  lActualYear + lTime));
}

double ANNUITYLV::dSetPre(double dValue)
{
  lValid = false;
  this->dPre = dValue;
  dPost = 1. - dPre;
  return(this->dPre);
}

double  ANNUITYLV::dSetRelativeQxForTime(long lTime, double dValue)
{
  lValid = false;
  return(psymRelQxTime->dSetValue(lTime, dValue));
}

void         ANNUITYLV::vLeistReset()
{
  double dTemp;
  long iC1;
  lValid = false;
  for (iC1 = 0; iC1 < 250; ++iC1)
    {
      dTemp = psymBenefit->dSetValue(iC1, 1.);
    }
}

void         ANNUITYLV::vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncre
{
  double dTemp = dStartValue, dTemp2;
  long iC1;
  lValid = false;
  for (iC1 = lTimeFrom; iC1 <= lTimeTo; ++iC1)
    {
      dTemp2 = psymBenefit->dSetValue(iC1, dTemp);
      dTemp += dIncrement;
    }
}
void         ANNUITYLV::vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor)
{
  double dTemp = dStartValue, dTemp2;
  long iC1;
  lValid = false;
  for (iC1 = lTimeFrom; iC1 <= lTimeTo; ++iC1)
    {
      dTemp2 = psymBenefit->dSetValue(iC1, dTemp);
```

```
        dTemp *= dFactor;
    }
}
```

Chapter 7

capital.h

```
//////////////////////////////////////////////////////////////////
//                                                              //
// Annuity LV Zahlungsstroeme                                   //
//                                                              //
// Autor Micahel Koller                                         //
//                                                              //
// Datum 3.2011: erstellt                                       //
//                                                              //
//////////////////////////////////////////////////////////////////

#ifndef  _OCAPITYL_INCLUDED
#define _OCAPITYL_INCLUDED

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include "omarkov.h"


class CAPITALLV:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  CAPITALLV();
  CAPITALLV(long lMaxTimesIpt); // Overrides Defaults
  ~CAPITALLV();
  int            iSetTable(char * pcId);
  void           vSetStartTime(long lTime);
  void           vSetStopTime(long lTime);
  void           vSetSurvival(long lTime, double dValue);
  void           vSetDeath(double dValue);
  void           vSetPremium(double dValue);
  void           vSetSurvivalGen(long lTime, double dValue); // nur einzelne Werte werden
  void           vSetDeathGen(long lTime, double dValue);                // ueberschrieben statt
  void           vSetPremiumGen(long lTime, double dValue);              // allen
  double         dSetQx(long lTime, double dValue);  // lNach irrelevant
  double         dSetFx(long lTime, double dValue);
  double         dSetBaseYear(long lTime);
  double         dSetActualYear(long lTime);
  double         dSetDisc(long lTime, double dValue);
  double         dGetDK(long lTime); // Berechnet DK's
  double         dGetCF(long lTime);
  double         dGetQx(long lTime, long lYear);
  double         dSetQx2Level(long lTime, double dValue);
  double         dSetSx2(long lTime, double dValue);
  double         dSetRDR(long lTime, double dValue);
```

```
  double          dSetSurenderPenaltyInMR(long lTime, double dValue);
  double          dSetSHMarginInMR(long lTime, double dValue);
  double          dSetSolaCapitalInMR(long lTime, double dValue);
  double          dSetInvReturn(long lTime, double dValue);
  double          dGetEV(long lTime);

private:
  long            lSAge;
  LV_VECTOR       *psymQx;
  LV_VECTOR       *psymFx;
  LV_VECTOR       *psymDisc;
  LV_VECTOR       *psymDeathBenefit;
  LV_VECTOR       *psymSurvivalBenefit;
  LV_VECTOR       *psymPremium;
  long            lBaseYear;
  long            lActualYear;
  bool            lValid;
  long            lMaxTime;
// For Embedded Value
  LV_VECTOR       *psymQx2Level;
  LV_VECTOR       *psymSx2;
  LV_VECTOR       *psymRDR;
  LV_VECTOR       *psymSurenderPenaltyInMR;
  LV_VECTOR       *psymSHMarginInMR;
  LV_VECTOR       *psymSolaCapitalInMR;
  LV_VECTOR       *psymInvReturn;
  MARKOVLV        *psymEV;
};

#endif
```

Chapter 8

capital.cpp

```
//////////////////////////////////////////////////////////////////
//                                                              //
// CAPITAL LV Zahlungsstroeme                                   //
//                                                              //
// Autor Michael Koller                                         //
//                                                              //
// Datum 3.2011: erstellt                                       //
//                                                              //
//////////////////////////////////////////////////////////////////
#include "capital.h"
#ifdef FOR_OLE
#pragma message ("In order to avoid double references to omarkov.cpp we include only header")
#include "omarkov.h"
#else
#include "omarkov.cpp"
#endif

CAPITALLV::CAPITALLV():MARKOVLV(250l,2l,1l)
{
  //  MARKOVLV::MARKOVLV(250l,2l,1l);
  vSetNrStates(2l);
  lSAge  =0l;
  psymQx = new LV_VECTOR(250, 0, 0);
  psymFx = new LV_VECTOR(250, 0, 0);
  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymDeathBenefit= new LV_VECTOR(250, 0, 0);
  psymSurvivalBenefit= new LV_VECTOR(250, 0, 0);
  psymPremium= new LV_VECTOR(250, 0, 0);
  lBaseYear = 2000l;
  lActualYear = 2000l;
  lValid = false;
  lMaxTime = 250;
  // For EV
  psymQx2Level             = new LV_VECTOR(250, 0, 0);
  psymSx2                  = new LV_VECTOR(250, 0, 0);
  psymRDR                  = new LV_VECTOR(250, 0, 0);
  psymSurenderPenaltyInMR  = new LV_VECTOR(250, 0, 0);
  psymSHMarginInMR         = new LV_VECTOR(250, 0, 0);
  psymSolaCapitalInMR      = new LV_VECTOR(250, 0, 0);
  psymInvReturn            = new LV_VECTOR(250, 0, 0);
  psymEV                   = new MARKOVLV(250, 3, 1); // Overrides Defaults;
}

CAPITALLV::CAPITALLV(long lMaxTimesIpt):MARKOVLV(250l,2l,1l)
{
  //  MARKOVLV::MARKOVLV(250l,3l,1l);
```

```
  vSetNrStates(3l);
  lSAge  =0l;
  psymQx = new LV_VECTOR(250, 0, 0);
  psymFx = new LV_VECTOR(250, 0, 0);
  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymDeathBenefit= new LV_VECTOR(250, 0, 0);
  psymSurvivalBenefit= new LV_VECTOR(250, 0, 0);
  psymPremium= new LV_VECTOR(250, 0, 0);
  lBaseYear = 2000l;
  lActualYear = 2000l;
  lValid = false;
  lMaxTime = lMaxTimesIpt;
  // For EV
  psymQx2Level             = new LV_VECTOR(250, 0, 0);
  psymSx2                  = new LV_VECTOR(250, 0, 0);
  psymRDR                  = new LV_VECTOR(250, 0, 0);
  psymSurenderPenaltyInMR  = new LV_VECTOR(250, 0, 0);
  psymSHMarginInMR         = new LV_VECTOR(250, 0, 0);
  psymSolaCapitalInMR      = new LV_VECTOR(250, 0, 0);
  psymInvReturn            = new LV_VECTOR(250, 0, 0);
  psymEV                   = new MARKOVLV(250, 3, 1); // Overrides Defaults;
}
CAPITALLV::~CAPITALLV()
{
  delete(psymQx);
  delete(psymFx);
  delete(psymDisc);
  delete(psymDeathBenefit);
  delete(psymSurvivalBenefit);
  delete(psymPremium);
  // For EV
  delete(psymQx2Level);
  delete(psymSx2);
  delete(psymRDR);
  delete(psymSurenderPenaltyInMR);
  delete(psymSHMarginInMR);
  delete(psymSolaCapitalInMR);
  delete(psymInvReturn);
  delete(psymEV);
}

int  CAPITALLV::iSetTable(char * pcId)
{
  int iC1,iC2;
  int iX0;
  int iXOmega;
  double dTech;
  lValid = false;
  psymQx->vReset();
  if (!this->psymTable1)
    this->psymTable1 = new TABLESERVER();
  this->psymTable1->vSetTable(pcId);
  iC2= this->psymTable1->iTableNumber();
  if (iC2<0)
    return(iC2);
  iX0 = this->psymTable1->iX0();
  iXOmega = this->psymTable1->iXOmega();
  vSetStartTime(iXOmega+1);
  dTech = this->psymTable1->dITech();
  for(iC1=0; iC1<2500; ++iC1)
    {
      dSetDisc(iC1, 1./(1.000000000001+ dTech));
    }
  for(iC1=0; iC1 <= iXOmega; ++iC1)
    {
```

```
      dSetQx(iC1, this->psymTable1->dGetValue(iC1));
    }
  return(iC2);
}

void CAPITALLV::vSetStartTime(long lTime)
{
  lValid = false;

  MARKOVLV::vSetStartTime(lTime);
  psymEV->vSetStartTime(lTime);
}

void CAPITALLV::vSetStopTime(long lTime)
{
  lValid = false;

  MARKOVLV::vSetStopTime(lTime);
  psymEV->vSetStopTime(lTime);
}

void CAPITALLV::vSetSurvival(long lTime, double dValue)
{
  int iC;
  lValid = false;
  for(iC=0; iC < 250;++iC)
    psymSurvivalBenefit->dSetValue(iC, 0.);
  if(lTime >= 1)
    psymSurvivalBenefit->dSetValue(lTime-1, dValue);
}
void CAPITALLV::vSetDeath(double dValue)
{
  int iC;
  lValid = false;
  for(iC=0; iC < 250;++iC)
    psymDeathBenefit->dSetValue(iC, dValue);
}

void  CAPITALLV::vSetPremium(double dValue)
{
  int iC;
  lValid = false;
  for(iC=0; iC < 250;++iC)
    psymPremium->dSetValue(iC, -dValue);
}

void CAPITALLV::vSetSurvivalGen(long lTime, double dValue)
{
  int iC;
  lValid = false;
  // Unterschied zu oben - es werden nicht alle anderen auf
  // Null gesetzt
  if(lTime >= 1)
    psymSurvivalBenefit->dSetValue(lTime-1, dValue);
}
void CAPITALLV::vSetDeathGen(long lTime, double dValue)
{
  lValid = false;
  // Unterschied zu oben nur ein Wert wird veraendert
    psymDeathBenefit->dSetValue(lTime, dValue);
}

void  CAPITALLV::vSetPremiumGen(long lTime, double dValue)
{
  lValid = false;
```

```
  // Unterschied zu oben nur ein Wert wird veraendert
    psymPremium->dSetValue(lTime, -dValue);
}

double CAPITALLV::dSetQx(long lTime, double dValue)
{
  lValid = false;
  return(psymQx->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetFx(long lTime, double dValue)
{
  lValid = false;
  return(psymFx->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetBaseYear(long lTime)
{
  lValid = false;
  lBaseYear = lTime;
  return((double) lTime);
}

double CAPITALLV::dSetActualYear(long lTime)
{
  lValid = false;
  lActualYear = lTime;
  return((double) lTime);
}

double CAPITALLV::dSetDisc(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 2500; ++ iC)
        dTemp = psymDisc->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymDisc->dSetValue(lTime, dValue));

}
double CAPITALLV::dGetDK(long lTime)
{
  int iC1, iC2= MARKOVLV::lGetStopTime();
  double dQxLoc, dSxLoc, dTemp, dLocDisc, dDKLocBoY,dDKLocEoY, dSHPartMR, dLocDisc2, dCostOfCapital;
  if (lValid == true) return(MARKOVLV::dGetDK(lTime,0l,1l));
  // Set Different Markov Elements before Calc
  lValid = true;
  lSAge = MARKOVLV::lGetStartTime();
  MARKOVLV::vReset();
  psymEV->vReset();
  psymEV->vSetNrStates(3);
  // A) For Mathematical Reserve
  // 1. Set Probabilities
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dQxLoc = dGetQx(iC1 , iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      dTemp = MARKOVLV::dSetPij(iC1, 0, 0, 1. - dQxLoc);
      dTemp = MARKOVLV::dSetPij(iC1, 0, 1, dQxLoc);
    }
  // 2. Set LV
  for(iC1=0; iC1<lSAge; ++iC1)
    {
```

```
      dTemp = MARKOVLV::dSetPre (iC1, 0, 0, psymPremium->dGetValue(iC1));
      dTemp = MARKOVLV::dSetPost(iC1, 0, 0, psymSurvivalBenefit->dGetValue(iC1));
      dTemp = MARKOVLV::dSetPost(iC1, 0, 1, psymDeathBenefit->dGetValue(iC1));
    }
  // 3. Set Discount
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dLocDisc = psymDisc->dGetValue(iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      dTemp = MARKOVLV::dSetDisc(iC1, 0, 0, dLocDisc);
      dTemp = MARKOVLV::dSetDisc(iC1, 1, 1, dLocDisc);
    }
  // B) For Embedded Value
  // 1. Set Probabilities
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dQxLoc = dGetQx(iC1 , iC1 - MARKOVLV::lGetStopTime() + lActualYear)*psymQx2Level->dGetValue(iC1);
      dSxLoc = psymSx2->dGetValue(iC1);
      dTemp = psymEV->dSetPij(iC1, 0, 0, 1. - dQxLoc - dSxLoc);
      dTemp = psymEV->dSetPij(iC1, 0, 1, dQxLoc);
      dTemp = psymEV->dSetPij(iC1, 0, 2, dSxLoc);
    }
  // 2. Set LV
  for(iC1=0; iC1<lSAge; ++iC1)
    {
      dDKLocBoY = MARKOVLV::dGetDK(iC1  ,0l,1l);
      dDKLocEoY = MARKOVLV::dGetDK(iC1+1,0l,1l);
      dLocDisc  = 1./psymDisc->dGetValue(iC1 - MARKOVLV::lGetStopTime() + lActualYear)-1.;
      dSHPartMR = dDKLocBoY * psymSHMarginInMR->dGetValue(iC1);
      dLocDisc2 = 1.+psymRDR->dGetValue(iC1);
      dCostOfCapital = psymSolaCapitalInMR->dGetValue(iC1)*dDKLocBoY*((1+psymInvReturn->dGetValue(iC1))
      // BoY Gets MR
      // EoY Pays MR + Benefits and gets Interest on MR BoY + SH_Part of additional return
      dTemp = psymEV->dSetPre  (iC1, 0, 0, dCostOfCapital + dDKLocBoY/dLocDisc2);
      dTemp = psymEV->dSetPost (iC1, 0, 0, dDKLocBoY*dLocDisc + dSHPartMR - dDKLocEoY - psymSurvivalBen
      dTemp = psymEV->dSetPost (iC1, 0, 1, dDKLocBoY*dLocDisc + dSHPartMR             - psymDeathBenefi
      dTemp = psymEV->dSetPost (iC1, 0, 2, dDKLocBoY*dLocDisc + dSHPartMR - dDKLocEoY + psymSurenderPen
    }
  // 3. Set Discount
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dLocDisc = 1./(1.+psymRDR->dGetValue(iC1));
      dTemp = psymEV->dSetDisc(iC1, 0, 0, dLocDisc);
      dTemp = psymEV->dSetDisc(iC1, 1, 1, dLocDisc);
      dTemp = psymEV->dSetDisc(iC1, 2, 2, dLocDisc);
    }

  // A) 4. Calc DK
  return(MARKOVLV::dGetDK(lTime,0l,1l));
}

double CAPITALLV::dGetCF(long lTime)
{
  double dTemp;
  if (lValid == false) dTemp = CAPITALLV::dGetDK(lTime);
  return(MARKOVLV::dGetCF(lTime, 0, 0) + MARKOVLV::dGetCF(lTime, 0, 1) );
}

double CAPITALLV::dGetQx(long lTime, long lYear)
{
  return(psymQx->dGetValue(lTime) * exp(psymFx->dGetValue(lTime) * (lYear - lBaseYear)));
}

double CAPITALLV::dSetQx2Level(long lTime, double dValue)
{
  lValid = false;
```

```
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymQx2Level->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymQx2Level->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetSx2(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymSx2->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymSx2->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetRDR(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymRDR->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymRDR->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetSurenderPenaltyInMR(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymSurenderPenaltyInMR->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymSurenderPenaltyInMR->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetSHMarginInMR(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymSHMarginInMR->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
```

```
    return(psymSHMarginInMR->dSetValue(lTime, dValue));
}
double CAPITALLV::dSetSolaCapitalInMR(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymSolaCapitalInMR->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymSolaCapitalInMR->dSetValue(lTime, dValue));
}

double CAPITALLV::dSetInvReturn(long lTime, double dValue)
{
  lValid = false;
  if(lTime < 0)
    {
      double dTemp;
      int iC;
      for(iC=0; iC < 250; ++ iC)
        dTemp = psymInvReturn->dSetValue(iC, dValue);
      return(dTemp);
    }
  else
    return(psymInvReturn->dSetValue(lTime, dValue));
}

double CAPITALLV::dGetEV(long lTime)
{
  double dTemp;
  if(lValid == false) dTemp = CAPITALLV::dGetDK(lTime);
  return(psymEV->dGetDK(lTime,0l,1l));
}
```

Chapter 9
annuity2.h

```
////////////////////////////////////////////////////////////////////
//                                                                //
// Annuity LV Zahlungsstroeme                                     //
//                                                                //
// Autor Michael Koller                                           //
//                                                                //
// Datum     3.2011: erstellt                                     //
//                                                                //
////////////////////////////////////////////////////////////////////

#ifndef  _OANNUITY2_INCLUDED
#define _OANNUITY2_INCLUDED

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include "omarkov.h"


class ANNUITYLV2:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  ANNUITYLV2();
  ANNUITYLV2(long lMaxTimesIpt, double dPre); // Overrides Defaults
  ~ANNUITYLV2();
  int            iSetTable1(char * pcId);
  int            iSetTable2(char * pcId);
  void           vSetStartTime(long lTime);
  void           vSetStopTime(long lTime);
  void           vSetSAge1(long lTime);
  void           vSetSAge2(long lTime);
  double         dSetQx1(long lTime, double dValue);  // lNach irrelevant
  double         dSetFx1(long lTime, double dValue);
  double         dSetQx2(long lTime, double dValue);  // lNach irrelevant
  double         dSetFx2(long lTime, double dValue);
  double         dSetBaseYear(long lTime);
  double         dSetActualYear(long lTime);
  double         dSetDisc(long lTime, double dValue);
  double         dGetDK(long lTime, long lState); // Berechnet DK's
  double         dGetCF(long lTime);
  double         dGetQx1(long lTime, long lYear);
  double         dGetQx2(long lTime, long lYear);
  double         dSetY_Minus_X(long lYAge, long lXAge);
  double         dSetBenefit(long lState, double dValue);
  double         dSetPre(double dValue);
```

```
  void          vLeistReset();
  void          vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncrement);
  void          vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor); // eac
private:
  long          lSAge1;
  long          lSAge2;
  LV_VECTOR     *psymQx1;
  LV_VECTOR     *psymFx1;
  LV_VECTOR     *psymQx2;
  LV_VECTOR     *psymFx2;
  LV_VECTOR     *psymDisc;
  LV_VECTOR     *psymBenefit;
  long          lBaseYear;
  long          lActualYear;
  bool          lValid;
  long          lMaxTime;
  double        dPreGen;
  double        dBenefit[3];
  double        dPre[3];
  double        dPost[3];
  long          lYMinusX;
};

#endif
```

Chapter 10
annuity2.cpp

```
////////////////////////////////////////////////////////////////////
//                                                                //
// Annuity LV Zahlungsstroeme (2 lives)                           //
//                                                                //
// Autor Michael Koller                                           //
//                                                                //
// Datum  3.2011: erstellt                                        //
//                                                                //
////////////////////////////////////////////////////////////////////
#include "annuity2.h"
#ifdef FOR_OLE
#pragma message ("In order to avoid double references to omarkov.cpp we include only header")
#include "omarkov.h"
#else
#include "omarkov.cpp"
#endif

ANNUITYLV2::ANNUITYLV2():MARKOVLV(250l,4l,1l)
{
  int iC;
  double dTemp;
  //  MARKOVLV::MARKOVLV(250l,4l,1l);
  vSetNrStates(4l);
  lSAge1  =0l;
  lSAge1  =0l;
  psymQx1 = new LV_VECTOR(250, 0, 0);
  psymFx1 = new LV_VECTOR(250, 0, 0);
  psymQx2 = new LV_VECTOR(250, 0, 0);
  psymFx2 = new LV_VECTOR(250, 0, 0);
  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymBenefit  = new LV_VECTOR(250, 0, 0);
  lBaseYear = 2000l;
  lActualYear = 2000l;
  lValid = false;
  lMaxTime = 250;
  dPreGen = 1;
  for(iC=0; iC<3; ++ iC)
    {
      this->dPre[iC] = dPreGen;
      dPost[iC] = 1. - dPre[iC];
      dBenefit[iC]  = 1.;
    }
  lYMinusX = 0;
  for (iC = 0; iC < 250; ++iC)
    {
      dTemp = psymBenefit->dSetValue(iC, 1.);
```

```
    }
}

ANNUITYLV2::ANNUITYLV2(long lMaxTimesIpt, double dPre):MARKOVLV(250l,4l,1l)
{
  int iC;
  double dTemp;
  //  MARKOVLV::MARKOVLV(250l,4l,1l);
  vSetNrStates(4l);
  lSAge1  =0l;
  lSAge1  =0l;
  psymQx1 = new LV_VECTOR(250, 0, 0);
  psymFx1 = new LV_VECTOR(250, 0, 0);
  psymQx2 = new LV_VECTOR(250, 0, 0);
  psymFx2 = new LV_VECTOR(250, 0, 0);
  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymBenefit  = new LV_VECTOR(250, 0, 0);
  lBaseYear = 2000l;
  lActualYear = 2000l;
  lValid = false;
  lMaxTime = lMaxTimesIpt;
  dPreGen = dPre;
  for(iC=0; iC<3; ++ iC)
    {
      this->dPre[iC] = dPreGen;
      this->dPost[iC] = 1. - this->dPre[iC];
      dBenefit[iC]  = 1.;
    }
  lYMinusX = 0;
  for (iC = 0; iC < 250; ++iC)
    {
      dTemp = psymBenefit->dSetValue(iC, 1.);
    }
}
ANNUITYLV2::~ANNUITYLV2()
{
  delete(psymQx1);
  delete(psymFx1);
  delete(psymQx2);
  delete(psymFx2);
  delete(psymDisc);
  delete(psymBenefit);
}

int  ANNUITYLV2::iSetTable1(char * pcId)
{
  int iC1,iC2;
  int iX0;
  int iXOmega;
  double dTech;
  lValid = false;
  psymQx1->vReset();
  if (!this->psymTable1)
    this->psymTable1 = new TABLESERVER();
  this->psymTable1->vSetTable(pcId);
  iC2= this->psymTable1->iTableNumber();
  if (iC2<0)
    return(iC2);
  iX0 = this->psymTable1->iX0();
  iXOmega = this->psymTable1->iXOmega();
  vSetStartTime(iXOmega+1);
  dTech = this->psymTable1->dITech();
  for(iC1=0; iC1<2500; ++iC1)
    {
      dSetDisc(iC1, 1./(1.000000000001+ dTech));
```

```
    }
  for(iC1=0; iC1 <= iXOmega; ++iC1)
    {
      dSetQx1(iC1, this->psymTable1->dGetValue(iC1));
    }
  return(iC2);
}

int  ANNUITYLV2::iSetTable2(char * pcId)
{
  int iC1,iC2;
  int iX0;
  int iXOmega;
  //  double dTech;
  lValid = false;
  psymQx2->vReset();
  if (!this->psymTable2)
    this->psymTable2 = new TABLESERVER();
  this->psymTable2->vSetTable(pcId);
  iC2= this->psymTable2->iTableNumber();
  if (iC2<0)
    return(iC2);
  iX0 = this->psymTable2->iX0();
  iXOmega = this->psymTable2->iXOmega();
  //  vSetStartTime(iXOmega+1);
  //dTech = this->psymTable1->dITech();
  //for(iC1=0; iC1<2500; ++iC1)
  //  {
  //    dSetDisc(iC1, 1./(1.000000000001+ dTech));
  //  }
  for(iC1=0; iC1 <= iXOmega; ++iC1)
    {
      dSetQx2(iC1, this->psymTable2->dGetValue(iC1));
    }
  return(iC2);
}

void ANNUITYLV2::vSetStartTime(long lTime)
{
  MARKOVLV::vSetStartTime(lTime);
}

void ANNUITYLV2::vSetStopTime(long lTime)
{
  MARKOVLV::vSetStopTime(lTime);
}
void ANNUITYLV2::vSetSAge1(long lTime)
{
  lValid = false;
  lSAge1  =lTime;
}
void ANNUITYLV2::vSetSAge2(long lTime)
{
  lValid = false;
  lSAge2  =lTime;
}
double ANNUITYLV2::dSetQx1(long lTime, double dValue)
{
  lValid = false;
  return(psymQx1->dSetValue(lTime, dValue));
}
double ANNUITYLV2::dSetFx1(long lTime, double dValue)
{
  lValid = false;
  return(psymFx1->dSetValue(lTime, dValue));
```

```
}
double ANNUITYLV2::dSetQx2(long lTime, double dValue)
{
  lValid = false;
  return(psymQx2->dSetValue(lTime, dValue));
}
double ANNUITYLV2::dSetFx2(long lTime, double dValue)
{
  lValid = false;
  return(psymFx2->dSetValue(lTime, dValue));
}
double ANNUITYLV2::dSetBaseYear(long lTime)
{
  lValid = false;
  lBaseYear = lTime;
  return((double) lTime);
}

double ANNUITYLV2::dSetActualYear(long lTime)
{
  lValid = false;
  lActualYear = lTime;
  return((double) lTime);
}

double ANNUITYLV2::dSetDisc(long lTime, double dValue)
{
  lValid = false;
  return(psymDisc->dSetValue(lTime, dValue));

}
double ANNUITYLV2::dGetDK(long lTime, long lState)
{
  int iC1, iC2= MARKOVLV::lGetStopTime();
  double dQxLoc1, dQxLoc2, dTemp, dLocDisc, dLeist;
  if (lValid == true) return(MARKOVLV::dGetDK(lTime,lState,1l));
  // Set Different Markov Elements before Calc
  lValid = true;
  MARKOVLV::vReset();
  // 1. Set Probabilities
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dQxLoc1 = dGetQx1(iC1 , iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      dQxLoc2 = dGetQx2(iC1 , iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      dTemp = MARKOVLV::dSetPij(iC1, 0, 0, (1. - dQxLoc1)*(1. - dQxLoc2));
      dTemp = MARKOVLV::dSetPij(iC1, 0, 1, (1. - dQxLoc1)* dQxLoc2);
      dTemp = MARKOVLV::dSetPij(iC1, 0, 2, dQxLoc1 * (1. - dQxLoc2));
      dTemp = MARKOVLV::dSetPij(iC1, 0, 3, dQxLoc1 * dQxLoc2 );
      dTemp = MARKOVLV::dSetPij(iC1, 1, 1, (1. - dQxLoc1));
      dTemp = MARKOVLV::dSetPij(iC1, 1, 3,  dQxLoc1);
      dTemp = MARKOVLV::dSetPij(iC1, 2, 2, (1. - dQxLoc2));
      dTemp = MARKOVLV::dSetPij(iC1, 2, 3, dQxLoc2);
    }
  // 2. Set LV
  for(iC1=lSAge1; iC1<lMaxTime; ++iC1)
    {
      dLeist = psymBenefit->dGetValue(iC1);
      dTemp = MARKOVLV::dSetPre(iC1, 0, 0, dLeist * dPre[0]);
      dTemp = MARKOVLV::dSetPost(iC1, 0, 0, dLeist * dPost[0]);
      dTemp = MARKOVLV::dSetPre(iC1, 1, 1, dLeist * dPre[1]);
      dTemp = MARKOVLV::dSetPost(iC1, 1, 1, dLeist * dPost[1]);
      dTemp = MARKOVLV::dSetPost(iC1, 0, 1, dLeist * dPost[1]);
    }
  for(iC1=lSAge2; iC1<lMaxTime; ++iC1)
    {
```

```
      dLeist = psymBenefit->dGetValue(iC1);
      dTemp = MARKOVLV::dSetPre(iC1, 2, 2, dLeist * dPre[2]);
      dTemp = MARKOVLV::dSetPost(iC1, 2, 2, dLeist * dPost[2]);
      dTemp = MARKOVLV::dSetPost(iC1, 0, 2, dLeist * dPost[2]);
    }
  // 3. Set Discount
  for(iC1=0; iC1<lMaxTime; ++iC1)
    {
      dLocDisc = psymDisc->dGetValue(iC1 - MARKOVLV::lGetStopTime() + lActualYear);
      dTemp = MARKOVLV::dSetDisc(iC1, 0, 0, dLocDisc);
      dTemp = MARKOVLV::dSetDisc(iC1, 1, 1, dLocDisc);
      dTemp = MARKOVLV::dSetDisc(iC1, 2, 2, dLocDisc);
      dTemp = MARKOVLV::dSetDisc(iC1, 3, 3, dLocDisc);
    }
  // 4. Calc DK
  return(MARKOVLV::dGetDK(lTime,lState,1l));
}

double ANNUITYLV2::dGetCF(long lTime)
{
  double dTemp;
  if (lValid == false) dTemp = ANNUITYLV2::dGetDK(lTime, 0);
  return(MARKOVLV::dGetCF(lTime, 0, 0) + MARKOVLV::dGetCF(lTime, 0, 1) + MARKOVLV::dGetCF(lTime, 0, 3)
);
}

double ANNUITYLV2::dGetQx1(long lTime, long lYear)
{
  return(psymQx1->dGetValue(lTime) * exp(psymFx1->dGetValue(lTime) * (lYear - lBaseYear)));
}
double ANNUITYLV2::dGetQx2(long lTime, long lYear)
{
  lTime += lYMinusX;  // Everything is calculated based on the age of the first life !!
  return(psymQx2->dGetValue(lTime) * exp(psymFx2->dGetValue(lTime) * (lYear - lBaseYear)));
}

double  ANNUITYLV2::dSetY_Minus_X(long lYAge, long lXAge)
{
  lValid = false;
  lYMinusX = lYAge - lXAge;
  return(lYMinusX);
}
double  ANNUITYLV2::dSetBenefit(long lState, double dValue)
{
  lValid = false;
  if(lState >= 0 && lState <= 2)
    {
      dBenefit[lState]  = dValue;
      this->dPre[lState] = dPreGen * dValue;
      dPost[lState] = dValue - dPre[lState];
      return(dValue);
    }
  else
    return(0.);
}

double ANNUITYLV2::dSetPre(double dValue)
{
int iC;
double dTemp;
dPreGen = dValue;
for(iC=0; iC<3 ; ++iC)
  {
  dTemp = this->dPre[iC] + this->dPost[iC];
  this->dPre[iC] = dPreGen * dValue;
```

```
  dPost[iC] = dValue - dPre[iC];
  }
return(this->dPreGen);
}

void         ANNUITYLV2::vLeistReset()
{
  double dTemp;
  long iC1;
  for (iC1 = 0; iC1 < 250; ++iC1)
    {
      dTemp = psymBenefit->dSetValue(iC1, 1.);
    }
}

void         ANNUITYLV2::vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncr
{
  double dTemp = dStartValue, dTemp2;
  long iC1;
  for (iC1 = lTimeFrom; iC1 <= lTimeTo; ++iC1)
    {
      dTemp2 = psymBenefit->dSetValue(iC1, dTemp);
      dTemp += dIncrement;
    }
}
void         ANNUITYLV2::vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor)
{
  double dTemp = dStartValue, dTemp2;
  long iC1;
  for (iC1 = lTimeFrom; iC1 <= lTimeTo; ++iC1)
    {
      dTemp2 = psymBenefit->dSetValue(iC1, dTemp);
      dTemp *= dFactor;
    }
}
```

Chapter 11

glmod.h

```
//////////////////////////////////////////////////////////////////
//                                                              //
// Annuity LV Zahlungsstroeme                                   //
//                                                              //
// Autor Michael Koller                                         //
//                                                              //
// Datum 3.2011: erstellt                                       //
//                                                              //
//////////////////////////////////////////////////////////////////

#ifndef  _OGLMOD_INCLUDED
#define _OGLMOD_INCLUDED

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include "omarkov.h"

#include "annuity.h"
#include "capital.h"

class GLMOD:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
 public:
  GLMOD();
  ~GLMOD();
  double          dSetQx(long lTable, long lType, long lSex, long lTime, double dValue);  // Table 0=K,
  double          dSetFx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetSx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetBaseYear(long lTable, long lType, long lSex, long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);

  void            vStress(long lType, double dAmount); // Idee lType = 0 -> Reset  1= = ...
  void            vAddAnnuity(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem, d
  void            vAddEndowment(long lSex, long lX, long lS, double dLeist, double dPraem, double dITech
  void            vAddWiddow(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem, do
  void            vSetRKWAnnuity(long lType, double dAmount);
  void            vSetRKWEndowment(long lType, double dAmount);
  void            vUpdateOperator();

  double          dGetDK(long lTime); // Berechnet DK's
  double          dGetDKDetail(long lTime, long lState); // Berechnet DK's fuer jeden State.
  double          dGetDKTilde(long lTime); // Berechnet DK's gesehen aus Zeit Null
  double          dGetStatDK(void);
```

```
  double          dGetFVDK(void);
  double          dGetCF(long lTime);
  double          dGetCFDetail(long lTime, long lState);
  double          dGetStatDK(long lType);  // lType: 0=Stat 1=FV ohne sx
  double          dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear);
  double          dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx
  int             iReadInforce(int iP, int iL, char * strFileName);
  void            vPrintTex(char * strName);
 private:
  bool            lValid;
  bool            bTildeCalc;
  LV_VECTOR       *psymQx[2][2][2];  // Tafel1/2 ; K oder R ; sex
  LV_VECTOR       *psymFx[2][2][2];
  LV_VECTOR       *psymDisc;
  LV_VECTOR       *psymSx;
  LV_VECTOR       *psymRelQxTime;
  LV_VECTOR       *psymTilde;
  long            lBaseYear[2][2][2];
  long            lActualYear;

  int         iQxStress;                          double dQxStress;    // qx -> (1 +/- alpha) x qx
  int         iFxStress;                          double dFxStress;    // qx -> (1 +/- alpha) x qx
  int         iSxStress;                          double dSxStress;    // sx -> dFxStress
  int         iYieldStress;                       double dYieldStress;   // yield -> yield +/- alpha

  int         iAexRKW;                            double dAexRKW;    // yield -> yield +/- alpha
  int         iAxRKW;                             double dAxRKW;    // yield -> yield +/- alpha

  double      dStatDK;
  double      dFVDK;

  FILE    *   psymTrace;
  ANNUITYLV   * psymAex;
  CAPITALLV   * psymPraem;
  CAPITALLV   * psymAx;

};

#endif
```

Chapter 12

glmod.cpp

```
//////////////////////////////////////////////////////////////////
//                                                              //
// GLMod LV Zahlungsstroeme                                     //
//                                                              //
// Autor Michael Koller                                         //
//                                                              //
// Datum 3.2011: erstellt                                       //
//                                                              //
//////////////////////////////////////////////////////////////////

#define PRO_MARKOV_STANDALONE

#ifdef PRO_MARKOV_STANDALONE
#define FOR_OLE
#include "annuity.h"
#include "capital.h"
#else
#include "annuity.cpp"
#include "capital.cpp"
#endif
#include "glmod.h"

#ifdef FOR_OLE
#pragma message ("In order to avoid double references to omarkov.cpp we include only header")
#include "omarkov.h"
#else
#include "omarkov.cpp"
#endif

#define TRACE_GLMOD
#define TRACE_PLUS
#define MAN               0l
#define WOMAN          1001l
#define CAPITAL           0l
#define ANNUITY        2001l
#define MAXAGE          991
#define SURRENDER      4001l
#define GLMDEATH       4011l
#define FIRSTORDER        0l
#define SECONDORDER       1l
#define KTAF              0l
#define RTAF              1l
#define STARTTIME      1201l
#define STOPTIME          0l
#define GLMNRSTATES    4021l
```

```
#define RESETSTRESS       0l
#define QXSTRESS          1l
#define FXSTRESS          2l
#define SXSTRESS          3l
#define YIELDSTRESS       4l

#define RKWNULL           0l  //Keine Rückkäufe eg sx=0
#define RKWDK             1l  //Rückkäufe alpha x DK

GLMOD::GLMOD():MARKOVLV(250l, GLMNRSTATES ,1l)
{
  int iC1, iC2, iC3;
  long lNTimes = 4;
  double dTemp;
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);


  //  MARKOVLV::MARKOVLV(250l, GLMNRSTATES ,1l);
  vSetNrStates(GLMNRSTATES);
  for(iC1= 0; iC1 < 2; ++iC1)
    {
      for(iC2= 0; iC2 < 2; ++iC2)
        {
          for(iC3= 0; iC3 < 2; ++iC3)
            {
              psymQx[iC1][iC2][iC3]    = new LV_VECTOR(250, 0, 0);
              psymFx[iC1][iC2][iC3]    = new LV_VECTOR(250, 0, 0);
              lBaseYear[iC1][iC2][iC3] = 2000l;
            }
        }
    }

  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymSx   = new LV_VECTOR(250, 0, 0);
  psymRelQxTime = new LV_VECTOR(250, 0, 0);
  psymTilde = new LV_VECTOR(250, 0, 0);
  lActualYear = 2000;

  iQxStress = 0;
  dQxStress = 0;   // qx -> (1 +/- alpha) x qx
  iFxStress = 0;
  dFxStress = 0;   // qx -> (1 +/- alpha) x qx
  iSxStress = 0;
  dSxStress = 0;   // sx -> dFxStress
  iYieldStress=0;
  dYieldStress=0;   // yield -> yield +/- alpha

  dStatDK =0;
  dFVDK   =0;
  bTildeCalc = false;
  lValid = false;
  dAddBenefits = true;

#ifdef TRACE_GLMOD
  psymTrace = fopen("trace_GLMOD.dat","w");
#else
  psymTrace = NULL;
#endif

  dAexRKW = 1.;
  dAxRKW  = 1.;
  //ANNUITYLV  * psymAex;
  //CAPITALLV  * psymPraem;
  psymAex = new ANNUITYLV(250, dPre); // Overrides Defaults
```

```
  psymPraem = new CAPITALLV(250); // Overrides Defaults
  psymAx = new CAPITALLV(250); // Overrides Defaults

  for (iC1 = 0; iC1 < 250; ++iC1) dTemp = psymRelQxTime->dSetValue(iC1, 1.);
}

GLMOD::~GLMOD()
{
  int iC1, iC2, iC3;
  for(iC1= 0; iC1 < 2; ++iC1)
    {
      for(iC2= 0; iC2 < 2; ++iC2)
        {
          for(iC3= 0; iC3 < 2; ++iC3)
            {
              delete(psymQx[iC1][iC2][iC3]);
              delete(psymFx[iC1][iC2][iC3]);
            }
        }
    }
  delete(psymDisc);
  delete(psymSx);
  delete(psymRelQxTime);
  delete(psymTilde);
#ifdef TRACE_GLMOD
  fclose(psymTrace);
#endif

  delete(psymAex);
  delete(psymPraem);
  delete(psymAx);
}

double          GLMOD::dSetQx(long lTable, long lType, long lSex, long lTime, double dValue)
// Table 0=K, 1= R Type 0 = 2 Ordn 1= erster Ordn
{
  if (lTable != 0 && lTable != 1) return(-1);
  if (lType  != 0 && lType != 1) return(-1);
  if (lSex   != 0 && lSex != 1) return(-1);
  lValid = false;
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\nQX>> Qx[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10.8f",
#endif
  return(psymQx[lTable][lType][lSex]->dSetValue(lTime, dValue));
}

double          GLMOD::dSetFx(long lTable, long lType, long lSex, long lTime, double dValue)
{
  if (lTable != 0 && lTable != 1) return(-1);
  if (lType  != 0 && lType  != 1) return(-1);
  if (lSex   != 0 && lSex   != 1) return(-1);
  lValid = false;
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\nFX>> Fx[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10.8f",
#endif
  return(psymFx[lTable][lType][lSex]->dSetValue(lTime, dValue));
}

double          GLMOD::dSetSx(long lTable, long lType, long lSex, long lTime, double dValue)
{
  // remark only one sx
  lValid = false;
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\nSX>> Sx[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10.8f",
#endif
```

```
  return(psymSx->dSetValue(lTime, dValue));

}

double          GLMOD::dSetBaseYear(long lTable, long lType, long lSex, long lTime)
{
  if (lTable != 0 && lTable != 1) return(-1);
  if (lType  != 0 && lType  != 1) return(-1);
  if (lSex   != 0 && lSex   != 1) return(-1);
  lValid = false;
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\nBY>> BaseYear[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10
#endif
  lBaseYear[lTable][lType][lSex]=lTime;
  return(lBaseYear[lTable][lType][lSex]);
}

double          GLMOD::dSetActualYear(long lTime)
{
  lValid = false;
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\nAY>> ActuarYear = %10ld", lTime);
#endif
  lActualYear = lTime;
  return(lActualYear);
}

double          GLMOD::dSetDisc(long lTime, double dValue)
{
  lValid = false;
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\nDI>> Disc[t=%4d] = %10.8lf",lTime, dValue);
#endif
  return(psymDisc->dSetValue(lTime, dValue));
}

void            GLMOD::vStress(long lType, double dValue) // Idee lType = 0 -> Reset  1= = ...
{
  if(lType == RESETSTRESS ) {iQxStress=0; iFxStress=0; iSxStress =0; iYieldStress=0;}
  if(lType == QXSTRESS    ) {iQxStress=1; dQxStress = dValue;}
  if(lType == FXSTRESS    ) {iFxStress=1; dFxStress = dValue;}
  if(lType == SXSTRESS    ) {iSxStress=1; dSxStress = dValue;}
  if(lType == YIELDSTRESS ) {iYieldStress=1; dYieldStress = dValue;}
  lValid = false;
}

void            GLMOD::vAddAnnuity(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPra
{
  double dTemp;
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);
  int iC1, iC2 =0;
  double dV = 1./(1.+dITechn);

  //ANNUITYLV  * psymAex;
  //CAPITALLV  * psymPraem;

  //psymAex = new ANNUITYLV(250, dPre); // Overrides Defaults
  //psymPraem = new CAPITALLV(250); // Overrides Defaults
  dTemp = psymAex->dSetPre(dPre);
  psymAex->vSetStartTime(STARTTIME);
  psymAex->vSetStopTime(lX);
  psymAex->vSetSAge(lS);
  psymAex->vSetG(0);
```

```
  psymPraem->vSetStartTime(lS);
  psymPraem->vSetStopTime(lX);
  psymPraem->vSetPremium(dPraem);

  for (iC1=lX; iC1 < STARTTIME; ++iC1)
    {
      psymAex->dSetQx(iC1, psymQx[FIRSTORDER][RTAF][lSex]->dGetValue(iC1));  // lNach irrelevant
      psymAex->dSetFx(iC1, psymFx[FIRSTORDER][RTAF][lSex]->dGetValue(iC1));
      psymAex->dSetDisc(lActualYear+iC1-lX, dV);

      psymPraem->dSetQx(iC1, psymQx[FIRSTORDER][RTAF][lSex]->dGetValue(iC1));  // lNach irrelevant
      psymPraem->dSetFx(iC1, psymFx[FIRSTORDER][RTAF][lSex]->dGetValue(iC1));
      psymPraem->dSetDisc(lActualYear+iC1-lX, dV);


    }
  psymAex->dSetBaseYear(lBaseYear[FIRSTORDER][RTAF][lSex]);
  psymAex->dSetActualYear(lActualYear);


  int iStateStar= WOMAN * lSex + ANNUITY +lX;

  if (lX <0 || lX > MAXAGE) { return;}

  dPre  *= dLeist;
  dPost *= dLeist;
  iC2=0;

  for(iC1 = lX; iC1 < lS; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre(iC2, iStateStar, iStateStar, -dPraem);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, SURRENDER, (psymAex->dGetDK(iC1) * dLeist + psymPraem
      ++iC2;
    }
  for(iC1 = lS; iC1 < STARTTIME; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre (iC2, iStateStar, iStateStar, dPre);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, iStateStar, dPost);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, SURRENDER, (psymAex->dGetDK(iC1) * dLeist + psymPraem
      ++iC2;
    }

  dStatDK += psymAex->dGetDK(lX) * dLeist + psymPraem->dGetDK(lX);
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\n vAddAnnuity  S %d, x %d SL %d L %10.4f P %10.4f iT %10.4f
DK %10.2f",lSex, lX, lS, dLeist, dPraem, dITechn, psymAex->dGetDK(lX) * dLeist + psymPraem->dGetDK(lX))
#endif
  for (iC1=lX; iC1 < STARTTIME; ++iC1)
    {
      double dTL = psymDisc->dGetValue(iC1);
      psymAex->dSetDisc(lActualYear+iC1-lX, dTL);
    }

  dFVDK += psymAex->dGetDK(lX) * dLeist + psymPraem->dGetDK(lX);
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace," FV %10.2f",psymAex->dGetDK(lX) * dLeist + psymPraem->dGetDK(lX));
#endif
  //  delete(psymAex);
  //  delete(psymPraem);

}

void           GLMOD::vAddEndowment(long lSex, long lX, long lS, double dLeist, double dPraem, double d
{
  double dTemp;
```

```
  double dV = 1./(1.+dITechn);
  int iC1, iC2 =0;

  // CAPITALLV  * psymAx;
  // psymAx = new CAPITALLV(250); // Overrides Defaults
  psymAx->vSetStartTime(lS);
  psymAx->vSetStopTime(lX);
  psymAx->vSetSurvival(lS, dLeist);
  psymAx->vSetDeath(dLeist);
  psymAx->vSetPremium(dPraem);


  for (iC1=lX; iC1 < STARTTIME; ++iC1)
    {
      psymAx->dSetQx(iC1, psymQx[FIRSTORDER][RTAF][lSex]->dGetValue(iC1));  // lNach irrelevant
      psymAx->dSetFx(iC1, psymFx[FIRSTORDER][RTAF][lSex]->dGetValue(iC1));
      psymAx->dSetDisc(lActualYear+iC1-lX, dV);
    }
  psymAx->dSetBaseYear(lBaseYear[FIRSTORDER][RTAF][lSex]);
  psymAx->dSetActualYear(lActualYear);


  int iStateStar= WOMAN * lSex + CAPITAL +lX;
  if (lX <0 || lX > MAXAGE) {return;}
  iC2=0;
  for(iC1 = lX; iC1 < lS; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre (iC2, iStateStar, iStateStar, -dPraem);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, GLMDEATH, dLeist);
      if(iC1 == lS - 1) dTemp = MARKOVLV::dSetPost(iC2, iStateStar, iStateStar, dLeist);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, SURRENDER, psymAx->dGetDK(iC1) * dAxRKW);
      ++iC2;
    }

  dStatDK += psymAx->dGetDK(lX);
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\n vAddCapital  S %d, x %d SL %d L %10.4f P %10.4f iT %10.8f DK %10.
#endif
  for (iC1=lX; iC1 < STARTTIME; ++iC1)
    {
      double dTL = psymDisc->dGetValue(iC1);
      psymAx->dSetDisc(lActualYear+iC1-lX, dTL);
    }

  dFVDK += psymAx->dGetDK(lX);
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace," FV %10.2f",psymAx->dGetDK(lX));
#endif
  //  delete(psymAx);

}

void           GLMOD::vAddWiddow(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPrae
{
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);
  double dTemp;
  double dV = 1./(1.+dITechn);
  int iC1, iC2 =0;
  int iDeltaXY = -3;

  if(lSex == 0){ iDeltaXY = -iDeltaXY;}

  dTemp = psymAex->dSetPre(dPre);
  psymAex->vSetStartTime(STARTTIME);
```

```
  psymAex->vSetStopTime(lX+iDeltaXY);
  psymAex->vSetSAge(0);
  psymAex->vSetG(0);

  for (iC1=lX; iC1 < STARTTIME; ++iC1)
    {
      psymAex->dSetQx(iC1, psymQx[FIRSTORDER][RTAF][1-lSex]->dGetValue(iC1));  // lNach irrelevant
      psymAex->dSetFx(iC1, psymFx[FIRSTORDER][RTAF][1-lSex]->dGetValue(iC1));
      psymAex->dSetDisc(lActualYear+iC1-lX, dV);
    }
  psymAex->dSetBaseYear(lBaseYear[FIRSTORDER][RTAF][1-lSex]);
  psymAex->dSetActualYear(lActualYear);


  int iStateStar= WOMAN * lSex + CAPITAL +lX;
  if (lX <0 || lX > MAXAGE) {return;}
  iC2=0;
  for(iC1 = lX; iC1 < lS; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre (iC2, iStateStar, iStateStar, -dPraem);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, GLMDEATH, dLeist*psymAex->dGetDK(iC1+iDeltaXY));
      // existiert nicht if(iC1 == lS - 1) dTemp = MARKOVLV::dSetPost(iC2, iStateStar, iStateStar, dLei
      // kein RKW dTemp = MARKOVLV::dSetPost(iC2, iStateStar, SURRENDER, psymAx->dGetDK(iC1) * dAxRKW);
      ++iC2;
    }

#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace,"\n vAddWiddow  #NV S %d, x %d SL %d L %10.4f P %10.4f iT %10.8f DK %
#endif
  for (iC1=lX; iC1 < STARTTIME; ++iC1)
    {
      double dTL = psymDisc->dGetValue(iC1);
      psymAx->dSetDisc(lActualYear+iC1-lX, dTL);
    }

  dFVDK += psymAx->dGetDK(lX);
#ifdef TRACE_GLMOD
  if(psymTrace) fprintf(psymTrace," FV %10.2f",psymAx->dGetDK(lX));
#endif
  //  delete(psymAx);

}

void           GLMOD::vSetRKWAnnuity(long lType, double dAmount)
{
  iAexRKW = lType;                              dAexRKW = dAmount;   // yield -> yield +/- alpha
}

void           GLMOD::vSetRKWEndowment(long lType, double dAmount)
{
  iAxRKW = lType;                               dAxRKW= dAmount;   // yield -> yield +/- alpha
}

void           GLMOD::vUpdateOperator()
{
  double dQxLoc, dSxLoc;
  int iStartalter, iVersicherungstyp, iC1, iC2;
  double dSxKorr = dSxStress;
  double dTemp;
  // 1. Belegen der Wahrscheinlichkeiten
  // ------------------------------------
  // Kapital Mann
  printf("\n Update Op");
  for(iStartalter = 0; iStartalter <= MAXAGE; ++ iStartalter)
    {
```

```
      for(iC1=0; iC1<STARTTIME; ++iC1)
        {
          dQxLoc = dGetQx(SECONDORDER, KTAF, MAN, iStartalter + iC1 , iC1 + lActualYear) * psymRelQxTim
          //printf("\n iC= %d x=%3d y=%4d qx = %10.8f Delta = %10.8f", iC1, iStartalter +iC1, iC1 + lAc
          dSxLoc = psymSx->dGetValue(iStartalter + iC1);
          if(iSxStress == 1 && iC1 == 0) dSxLoc = dSxKorr;
          //      printf("\n iC= %d x=%3d y=%4d sx = %10.8f", iC1, iStartalter +iC1, iC1 + lActualYear,
          dTemp = MARKOVLV::dSetPij(iC1, MAN + CAPITAL + iStartalter , MAN + CAPITAL + iStartalter, (1.
          dTemp = MARKOVLV::dSetPij(iC1, MAN + CAPITAL + iStartalter , GLMDEATH, (1. - dSxLoc) * dQxLoc
          dTemp = MARKOVLV::dSetPij(iC1, MAN + CAPITAL + iStartalter , SURRENDER, dSxLoc);
        }


    }
  // Kapital Frau
  for(iStartalter = 0; iStartalter <= MAXAGE; ++ iStartalter)
    {
      for(iC1=0; iC1<STARTTIME; ++iC1)
        {
          dQxLoc = dGetQx(SECONDORDER, KTAF, 1 /* for woman */, iStartalter + iC1 , iC1 + lActualYear)*
          dSxLoc = psymSx->dGetValue(iStartalter + iC1);
          if(iSxStress == 1 && iC1 == 0) dSxLoc = dSxKorr;
          dTemp = MARKOVLV::dSetPij(iC1, WOMAN + CAPITAL + iStartalter , WOMAN + CAPITAL + iStartalter,
          dTemp = MARKOVLV::dSetPij(iC1, WOMAN + CAPITAL + iStartalter , GLMDEATH, (1. - dSxLoc) * dQxL
          dTemp = MARKOVLV::dSetPij(iC1, WOMAN + CAPITAL + iStartalter , SURRENDER, dSxLoc);
        }

    }
  // Rente Mann
  for(iStartalter = 0; iStartalter <= MAXAGE; ++ iStartalter)
    {
      for(iC1=0; iC1<STARTTIME; ++iC1)
        {
          dQxLoc = dGetQx(SECONDORDER, RTAF, MAN, iStartalter + iC1 , iC1 + lActualYear)* psymRelQxTime
          dSxLoc = psymSx->dGetValue(iStartalter + iC1);
          if(iSxStress == 1 && iC1 == 0) dSxLoc = dSxKorr;
          dTemp = MARKOVLV::dSetPij(iC1, MAN + ANNUITY + iStartalter , MAN + ANNUITY + iStartalter, (1.
          dTemp = MARKOVLV::dSetPij(iC1, MAN + ANNUITY + iStartalter , GLMDEATH, (1. - dSxLoc) * dQxLoc
          dTemp = MARKOVLV::dSetPij(iC1, MAN + ANNUITY + iStartalter , SURRENDER, dSxLoc);
        }

    }
  // Rente Frau
  for(iStartalter = 0; iStartalter <= MAXAGE; ++ iStartalter)
    {
      for(iC1=0; iC1<STARTTIME; ++iC1)
        {
          dQxLoc = dGetQx(SECONDORDER, RTAF, 1 /*woman*/, iStartalter + iC1 , iC1 + lActualYear)* psymR
          dSxLoc = psymSx->dGetValue(iStartalter + iC1);
          if(iSxStress == 1 && iC1 == 0) dSxLoc = dSxKorr;
          dTemp = MARKOVLV::dSetPij(iC1, WOMAN + ANNUITY + iStartalter , WOMAN + ANNUITY + iStartalter,
          dTemp = MARKOVLV::dSetPij(iC1, WOMAN + ANNUITY + iStartalter , GLMDEATH, (1. - dSxLoc) * dQxL
          dTemp = MARKOVLV::dSetPij(iC1, WOMAN + ANNUITY + iStartalter , SURRENDER, dSxLoc);
        }
    }
  // 2. Belegen der Zinsen
  // -------------------------------------
  for(iC1=0; iC1<STARTTIME; ++iC1)
    {
      double dLocDisc;
      dLocDisc = psymDisc->dGetValue(iC1);
      //    printf("\ t=%d disc = %10.8f ", iC1, dLocDisc);
      if(dLocDisc != 0 && iYieldStress)
        {
          dLocDisc  = (1./dLocDisc + dYieldStress);
```

```
          if (dLocDisc >= 1.) dLocDisc = 1./ dLocDisc;
          else                dLocDisc = 1.;
        }
      for(iC2=0; iC2 < 402; ++ iC2) dTemp = MARKOVLV::dSetDisc(iC1, iC2, iC2, dLocDisc);
    }
  lValid = false;
}

double          GLMOD::dGetStatDK(void)
{
  return(dStatDK);
}

double          GLMOD::dGetFVDK(void)
{
  return(dFVDK);
}


double       GLMOD::dGetDK(long lTime)
{
  double dTemp=0;
  int iC1;

  if(!lValid)
    {
      vUpdateOperator();

      MARKOVLV::vSetStartTime(STARTTIME);
      MARKOVLV::vSetStopTime(STOPTIME);
      lValid = true;
    }
  // printf("\n Get GLM DK");

  for(iC1= 0; iC1 < GLMNRSTATES; ++ iC1) dTemp += MARKOVLV::dGetDK(lTime,iC1,1l);
  return(dTemp);
}

double   GLMOD::dGetDKDetail(long lTime, long lState) // Berechnet DK's fuer jeden State.
{
  double dTemp=0;
  int iC1;

  if(!lValid)
    {
      dTemp = dGetDK(0);
    }
  return(MARKOVLV::dGetDK(lTime,lState,1l));
}

double          GLMOD::dGetCF(long lTime)
{
  double dTemp;
  int iC1;
  if(!lValid) dTemp = dGetDK(0);
  dTemp = 0.;
  for(iC1= 0; iC1 < GLMNRSTATES; ++ iC1) dTemp += MARKOVLV::dGetCF(lTime, iC1, iC1) + MARKOVLV::dGetCF(
  return(dTemp);
}

double          GLMOD::dGetCFDetail(long lTime, long lState)
{
  double dTemp;
  int iC1;
  if(!lValid) dTemp = dGetDK(0);
```

```
  return(MARKOVLV::dGetCF(lTime, lState, lState));
}


double       GLMOD::dGetDKTilde(long lTime)
{
  double dTemp=0;
  double dT2;
  int iC1;

  if (!bTildeCalc || !lValid)
    {
      for(iC1= STARTTIME -1; iC1 >= STOPTIME; -- iC1)
        {
          dTemp = dTemp * psymDisc->dGetValue(iC1) + dGetCF(iC1);
          dT2= psymTilde->dSetValue(iC1, dTemp);
        }
    bTildeCalc = true;
    }

  return(psymTilde->dGetValue(iC1));
}


double         GLMOD::dGetStatDK(long lType)  // lType: 0=Stat 1=FV ohne sx
{
  if(lType == 0)       return(dStatDK);
  if(lType == 1)       return(dFVDK);

  return(0.);
}

double GLMOD::dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear)
{
  long lDeltaT = lYear - lBaseYear[lOrder][lTafel][lSex];
  double dQxKorr = 1.;
  double dFxKorr = 1.;
  double dTemp;
  //if (lDeltaT > lMaxProj) lDeltaT = lMaxProj;
  if (iQxStress == 1) dQxKorr = 1 + dQxStress;
  if (iFxStress == 1) dFxKorr = 1 + dFxStress;
  dTemp = dQxKorr * psymQx[lOrder][lTafel][lSex]->dGetValue(lTime) * exp(dFxKorr * psymFx[lOrder][lTafe
  if(dTemp <0) dTemp =0.;
  if(dTemp >1) dTemp =1.;
  return(dTemp);
}

double  GLMOD::dSetRelativeQxForTime(long lTime, double dValue)
{
  lValid = false;
  //printf("\n Add Qx Time %5d %10lf",lTime, psymRelQxTime->dSetValue(lTime, dValue));
  return(psymRelQxTime->dSetValue(lTime, dValue));
}

int GLMOD::iReadInforce(int iP, int iL, char * strFileName)
{
  // Set Inforce File [File]
  int iMsg = 0;
  FILE * psymFile;
  char strBuffer[1024];
  char * pcChar;
  long lNrLines = 0;

  psymFile = fopen(strFileName,"r");
```

```
  if (psymFile == NULL)
    {
      return(1);
    }

  while(!feof(psymFile))
    {
      if(fgets(strBuffer,1024,psymFile))
        {
          // STRUKTUR        Tarif (A/G)  Geschlecht(1/2) Alter Schlussalter  Leistung Prämie Technisch
          //   void          vAddAnnuity(long lSex, long lX, long lS, long lNTimes, double dLeist, doub
          //   void          vAddEndowment(long lSex, long lX, long lS, double dLeist, double dPraem, d
          int iAge, iSex, iSA;
          char strTar[50];
          double dL, dP, dTi;
          pcChar = strBuffer;
          while(pcChar)
            {
              if(*pcChar != ' ') break;
              ++pcChar;
            }
          sscanf(pcChar,"%s %d %d %d %lf %lf %lf", strTar, &iSex, &iAge, &iSA,  &dL, &dP, &dTi);
#ifdef TRACE_GLMOD
          printf("\n Step 1: %s %d %d %d %lf %lf %lf", strTar, iSex, iAge, iSA,  dL, dP, dTi);
#else
          if(!(lNrLines % 100)) printf("\n Read %4d Lines",lNrLines);
#endif
          --iSex; if(iSex >=0 || iSex <=1)
                    {
                      if(strstr(strTar,"G")) vAddEndowment(iSex, iAge, iSA, iL*dL, iP*dP, dTi);
                      if(strstr(strTar,"A")) vAddAnnuity(iSex, iAge, iSA, 4, iL*dL, iP*dP, dTi);
                      if(strstr(strTar,"W")) vAddWiddow(iSex, iAge, iSA, 4, iL*dL, iP*dP, dTi);
                    }
          ++lNrLines;
        }
    }
  iMsg= fclose(psymFile);
  return(0);
  printf("\n Done: read of inforce %5d", lNrLines);
}


void   GLMOD::vPrintTex(char * strName)
{
  FILE * psymFile;
  printf("\n Start Printing Tex File");
  psymFile = fopen(strName,"w");
  if (psymFile != NULL)
    MARKOVLV::vPrintTeX(psymFile, true, "GLM TRACE", true);
  else
    printf("\n Open Tex File failed \n \n");
  printf("\n Tex Output to %s \n > done \n", strName);
  fclose(psymFile);
}
```

Chapter 13

annmod.h

```
//////////////////////////////////////////////////////////////////
//                                                              //
// Annuity LV Zahlungsstroeme                                   //
//                                                              //
// Autor Michael Koller                                         //
//                                                              //
// Datum 3.2011: erstellt                                       //
//                                                              //
//////////////////////////////////////////////////////////////////

#ifndef  _OANMOD_INCLUDED
#define _OANMOD_INCLUDED

#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <memory.h>
#include "omarkov.h"


#include "annuity.h"
#include "capital.h"

class ANNMOD:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
 public:
  ANNMOD();
  ~ANNMOD();
  double          dSetQx(long lTable, long lType, long lSex, long lTime, double dValue);  // Table 0=K,
  double          dSetFx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetSx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetBaseYear(long lTable, long lType, long lSex, long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);

  // Einlebig
  void            vAddAnnuity1(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem,
  // Zinsrente
  void            vAddAnnuity0(long lX, long lS, long lNTimes, double dLeist, double dPraem, double dITe
  // Zweilebig joint life
  void            vAddAnnuity2xy(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, double
  // Zweilebig last life
  void            vAddAnnuity2xyBar(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, dou
  // Zweilebig y widdow annuity
  void            vAddAnnuity2xToy(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, doub
  // Zweilebig y widdow annuity
```

```
  void           vAddAnnuity2yTox(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, doub


  void           vUpdateOperator();

  double         dGetDK(long lTime); // Berechnet DK's
  double         dGetStatDK(void);
  double         dGetFVDK(void);
  double         dGetCF(long lTime);
  double         dGetStatDK(long lType);  // lType: 0=Stat 1=FV ohne sx

  double         dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear);
  double         dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx

 private:
  bool           lValid;

  LV_VECTOR      * psymQx[2][2][2];  // Tafel1/2 ; K oder R ; sex
  LV_VECTOR      * psymFx[2][2][2];
  LV_VECTOR      * psymDisc;
  LV_VECTOR      * psymSx;
  LV_VECTOR      * psymRelQxTime;
  long           lBaseYear[2][2][2];
  long           lActualYear;

  double     dStatDK;
  double     dFVDK;

  FILE       * psymTrace;
  ANNUITYLV  * psymAex;
  CAPITALLV  * psymPraem;
  CAPITALLV  * psymAx;

};

#endif
```

Chapter 14

annmod.cpp

```
//////////////////////////////////////////////////////////////////
//                                                              //
// ANNMOD LV Zahlungsstroeme                                    //
//                                                              //
// Autor Michael Koller                                         //
//                                                              //
// Datum 3.2011: erstellt                                       //
//                                                              //
//////////////////////////////////////////////////////////////////
#include "annmod.h"

#ifdef FOR_OLE
#pragma message ("In order to avoid double references to omarkov.cpp we include only header")
#include "omarkov.h"
#else
#include "omarkov.cpp"
#endif

#define TRACE_ANNMOD
#define MAN              0l    // Maenner starten bei 0 (bis 99)
#define WOMAN          100l    // Maenner starten bei 100 (bis 199)
#define ONELIFE          0l    // Einlebige starten bei 0
#define TWOLIFE__XX    200l    // (x,x) Paare bei 200
#define TWOLIFE__XY    300l    // (x,y) Paare bei 300
#define TWOLIFE__YY    400l    // (y,y) Paare bei 400
#define DELTAXYM10       0l    // Offest x-y = -10
#define DELTAXYM3      300l    // Offest x-y = -3
#define DELTAXYP3      600l    // Offest x-y = +3
#define DELTAXYP8      900l    // Offest x-y = +8
#define DELTAXYP13    1200l    // Offest x-y = +13
#define MAXAGE          99l
#define ZINSRENTE     1700l    // In Zustand 1700 Zinsrente
#define ANNDEATH      1701l    // In Zustand 1701 Tod
#define FIRSTORDER       0l
#define SECONDORDER      1l
#define KTAF             0l
#define RTAF             1l
#define STARTTIME      120l
#define STOPTIME         0l
#define ANNNRSTATES   1702l


ANNMOD::ANNMOD():MARKOVLV(250l, ANNNRSTATES ,1l)
{
  //TODOS: a) Steigerung expo für alle renten
  int iC1, iC2, iC3;
```

```
  long lNTimes = 4;
  double dTemp;
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);

  //  MARKOVLV::MARKOVLV(250l, ANNNRSTATES ,1l);
  vSetNrStates(ANNNRSTATES);
  for(iC1= 0; iC1 < 2; ++iC1)
    {
      for(iC2= 0; iC2 < 2; ++iC2)
        {
          for(iC3= 0; iC3 < 2; ++iC3)
            {
              psymQx[iC1][iC2][iC3]    = new LV_VECTOR(250, 0, 0);
              psymFx[iC1][iC2][iC3]    = new LV_VECTOR(250, 0, 0);
              lBaseYear[iC1][iC2][iC3] = 2000l;
            }
        }
    }

  psymDisc = new LV_VECTOR(2500, 0, 0);
  psymSx   = new LV_VECTOR(250, 0, 0);
  psymRelQxTime = new LV_VECTOR(250, 0, 0);
  lActualYear = 2000;

  dStatDK =0;
  dFVDK   =0;
  lValid = false;
  dAddBenefits = true;

#ifdef TRACE_ANNMOD
  psymTrace = fopen("trace_ANNMOD.dat","w");
#else
  psymTrace = NULL;
#endif

  psymAex = new ANNUITYLV(250, dPre); // Overrides Defaults
  psymPraem = new CAPITALLV(250); // Overrides Defaults
  psymAx = new CAPITALLV(250); // Overrides Defaults

  for (iC1 = 0; iC1 < 250; ++iC1) dTemp = psymRelQxTime->dSetValue(iC1, 1.);
}

ANNMOD::~ANNMOD()
{
  int iC1, iC2, iC3;
  for(iC1= 0; iC1 < 2; ++iC1)
    {
      for(iC2= 0; iC2 < 2; ++iC2)
        {
          for(iC3= 0; iC3 < 2; ++iC3)
            {
              delete(psymQx[iC1][iC2][iC3]);
              delete(psymFx[iC1][iC2][iC3]);
            }
        }
    }
  delete(psymDisc);
  delete(psymSx);
  delete(psymRelQxTime);
#ifdef TRACE_ANNMOD
  fclose(psymTrace);
#endif

  delete(psymAex);
```

```
  delete(psymPraem);
  delete(psymAx);


}


double          ANNMOD::dSetQx(long lTable, long lType, long lSex, long lTime, double dValue)
                                    // Table 0=K, 1= R Type 0 = 2 Ordn 1= erster Ordn
{
  if (lTable != 0 && lTable != 1) return(-1);
  if (lType  != 0 && lType != 1) return(-1);
  if (lSex   != 0 && lSex != 1) return(-1);
  lValid = false;
#ifdef TRACE_ANNMOD
  if(psymTrace) fprintf(psymTrace,"\nQX>> Qx[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10.8f",
#endif
  return(psymQx[lTable][lType][lSex]->dSetValue(lTime, dValue));

}

double          ANNMOD::dSetFx(long lTable, long lType, long lSex, long lTime, double dValue)
{
  if (lTable != 0 && lTable != 1) return(-1);
  if (lType  != 0 && lType  != 1) return(-1);
  if (lSex   != 0 && lSex   != 1) return(-1);
  lValid = false;
#ifdef TRACE_ANNMOD
  if(psymTrace) fprintf(psymTrace,"\nFX>> Fx[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10.8f",
#endif
  return(psymFx[lTable][lType][lSex]->dSetValue(lTime, dValue));
}

double          ANNMOD::dSetSx(long lTable, long lType, long lSex, long lTime, double dValue)
{
  // remark only one sx
  lValid = false;
#ifdef TRACE_ANNMOD
  if(psymTrace) fprintf(psymTrace,"\nSX>> Sx[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10.8f",
#endif
  return(psymSx->dSetValue(lTime, dValue));

}

double          ANNMOD::dSetBaseYear(long lTable, long lType, long lSex, long lTime)
{
  if (lTable != 0 && lTable != 1) return(-1);
  if (lType  != 0 && lType  != 1) return(-1);
  if (lSex   != 0 && lSex   != 1) return(-1);
  lValid = false;
#ifdef TRACE_ANNMOD
  if(psymTrace) fprintf(psymTrace,"\nBY>> BaseYear[Table(K/R)=%1d][Typ(1/2Ord)=%1d][Sex(0/1)=%1d] = %10
#endif
  lBaseYear[lTable][lType][lSex]=lTime;
  return(lBaseYear[lTable][lType][lSex]);
}

double          ANNMOD::dSetActualYear(long lTime)
{
  lValid = false;
#ifdef TRACE_ANNMOD
  if(psymTrace) fprintf(psymTrace,"\nAY>> ActuarYear = %10ld", lTime);
#endif
  lActualYear = lTime;
  return(lActualYear);
```

```
}

double          ANNMOD::dSetDisc(long lTime, double dValue)
{
  lValid = false;
#ifdef TRACE_ANNMOD
  if(psymTrace) fprintf(psymTrace,"\nDI>> Disc[t=%4d] = %10.8lf",lTime, dValue);
#endif
  return(psymDisc->dSetValue(lTime, dValue));
}

// Einlebig
void            ANNMOD::vAddAnnuity1(long lSex, long lX, long lS, long lNTimes, double dLeist, double dP
{
  double dTemp;
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);
  int iC1, iC2 =0;
  double dV = 1./(1.+dITechn);
  double dBeta;

  int iStateStar= WOMAN * lSex + ONELIFE +lX;

  if (lX <0 || lX > MAXAGE) { return;}

  dPre  *= dLeist;
  dPost *= dLeist;
  iC2=0;

  for(iC1 = lX; iC1 < lS; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre(iC2, iStateStar, iStateStar, -dPraem);
      ++iC2;
    }

  dBeta = 1.;
  for(iC1 = lS; iC1 < STARTTIME; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre (iC2, iStateStar, iStateStar, dPre  * dBeta);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, iStateStar, dPost * dBeta);
      ++iC2; dBeta *= 1+ dIncrease;
    }
}

// Zinsrente
void            ANNMOD::vAddAnnuity0(long lX, long lS, long lNTimes, double dLeist, double dPraem, doubl
{
  double dTemp;
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);
  int iC1, iC2 =0;
  double dV = 1./(1.+dITechn);
  double dBeta;

  int iStateStar= ZINSRENTE;


  dPre  *= dLeist;
  dPost *= dLeist;
  iC2=0;

  dBeta = 1;
  for(iC1 = lX; iC1 < lS; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre (iC2, iStateStar, iStateStar, dPre * dBeta);
```

```
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, iStateStar, dPost * dBeta);
      ++iC2; dBeta *= 1+ dIncrease;
    }
}

// Zweilebig joint life
void           ANNMOD::vAddAnnuity2xy(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes,
{
  double dTemp;
  double dPre = (lNTimes + 1) /( 2. * lNTimes);
  double dPost= (lNTimes - 1) /( 2. * lNTimes);
  long   lDeltaXY;
  int iC1, iC2 =0;
  double dV = 1./(1.+dITechn);
  double dBeta;

  int iStateStar= lX;

  if (lX <0 || lX > MAXAGE) { return;}

  if (lSex1 == WOMAN && lSex2 == WOMAN)
    {
      iStateStar+= TWOLIFE__XX;
    }

  if (lSex1 == WOMAN && lSex2 == MAN)
    {
      long lTemp = lX;
      lS += lY - lX;
      lX = lY;
      lY = lTemp;
      lSex1 = MAN;
      lSex2 = WOMAN;
    }

  if (lSex1 == MAN && lSex2 == WOMAN)
    {
      iStateStar+= TWOLIFE__XY;
    }

  if (lSex1 == MAN && lSex2 == MAN)
    {
      iStateStar += TWOLIFE__YY;
    }

  // DELTAXY-10        0l    ]-infty , -7]
  // DELTAXY-3       300l    ]-7     , 0]
  // DELTAXY+3       600l    ]0      , 5]
  // DELTAXY+8       900l    ]5      , 10]
  // DELTAXY+13XY   1200l    ]10     , infty]

  lDeltaXY = lX - lY;
  // NOP if( lDeltaXY > -2000 && lDeltaXY <= -7)
  if( lDeltaXY > -7 && lDeltaXY <=  0) iStateStar += DELTAXYM3;
  if( lDeltaXY >  0 && lDeltaXY <=  5) iStateStar += DELTAXYP3;
  if( lDeltaXY >  5 && lDeltaXY <= 10) iStateStar += DELTAXYP8;
  if( lDeltaXY > 10                  ) iStateStar += DELTAXYP13;


  dPre  *= dLeist;
  dPost *= dLeist;
  iC2=0;

  for(iC1 = lX; iC1 < lS; ++ iC1)
    {
```

```
      dTemp = MARKOVLV::dSetPre(iC2, iStateStar, iStateStar, -dPraem);
      ++iC2;
    }

  dBeta = 1.;
  for(iC1 = lS; iC1 < STARTTIME; ++ iC1)
    {
      dTemp = MARKOVLV::dSetPre (iC2, iStateStar, iStateStar, dPre * dBeta);
      dTemp = MARKOVLV::dSetPost(iC2, iStateStar, iStateStar, dPost * dBeta);
      ++iC2; dBeta *= 1+ dIncrease;
    }
}

// Zweilebig last life
void           ANNMOD::vAddAnnuity2xyBar(long lSex1, long lSex2, long lX, long lY, long lS, long lNTime
{
  vAddAnnuity1(lSex1, lX, lS, lNTimes, dLeist, dPraem, dITechn, dIncrease);
  vAddAnnuity1(lSex2, lY, lS -lX + lY, lNTimes, dLeist, dPraem, dITechn, dIncrease);
  vAddAnnuity2xy(lSex1, lSex2, lX, lY, lS, lNTimes, -dLeist, -dPraem, dITechn, dIncrease);
}

// Zweilebig y widdow annuity
void           ANNMOD::vAddAnnuity2xToy(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes
{
  vAddAnnuity1(lSex2, lY, lS -lX + lY, lNTimes, dLeist, dPraem, dITechn, dIncrease);
  vAddAnnuity2xy(lSex1, lSex2, lX, lY, lS, lNTimes, -dLeist, -dPraem, dITechn, dIncrease);
}

// Zweilebig y widdow annuity
void           ANNMOD::vAddAnnuity2yTox(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes
{
  vAddAnnuity1(lSex1, lX, lS, lNTimes, dLeist, dPraem, dITechn, dIncrease);
  vAddAnnuity2xy(lSex1, lSex2, lX, lY, lS, lNTimes, -dLeist, -dPraem, dITechn, dIncrease);
}

void           ANNMOD::vUpdateOperator()
{
  double dQxLoc, dQyLoc;
  int iStartalter,  iC1, iC2, iC3, iYAge;
  double dTemp;
  int iShift[] ={ DELTAXYM10, DELTAXYM3, DELTAXYP3, DELTAXYP8, DELTAXYP13, 9999};
  int iDXY[]   ={ -10, -3, 3, 8, 13, 0};


  // DELTAXY-10        0l    ]-infty , -7]
  // DELTAXY-3       300l    ]-7     , 0]
  // DELTAXY+3       600l    ]0      , 5]
  // DELTAXY+8       900l    ]5      , 10]
  // DELTAXY+13XY   1200l    ]10     , infty]
  // TWOLIFE__XX     200l
  // TWOLIFE__XY     300l
  // TWOLIFE__YY     400l


  // 1. Belegen der Wahrscheinlichkeiten
  // -----------------------------------
  // Kapital Mann
  printf("\n Update Op");
  // Rente Mann
  for(iStartalter = 0; iStartalter <= MAXAGE; ++ iStartalter)
    {
      for(iC1=0; iC1<STARTTIME; ++iC1)
        {
          dQxLoc = dGetQx(SECONDORDER, RTAF, MAN, iStartalter + iC1 , iC1 + lActualYear)* psymRelQxTime
          dTemp = MARKOVLV::dSetPij(iC1, MAN + iStartalter , MAN + iStartalter, (1. - dQxLoc));
```

```
          dTemp = MARKOVLV::dSetPij(iC1, MAN + iStartalter , ANNDEATH,  dQxLoc);
          for (iC3=0; iShift[iC3] < 100; ++iC3)
            {
              iYAge =  iStartalter + iC1 + iDXY[iC3];
              if(iYAge < 0) iYAge = 0;
              // now m -> m
              dQyLoc = dGetQx(SECONDORDER, RTAF, MAN, iYAge , iC1 + lActualYear)* psymRelQxTime->dGetVa
              dTemp = MARKOVLV::dSetPij(iC1, TWOLIFE__XX + iStartalter + iShift[iC3] , TWOLIFE__XX + iS
              dTemp = MARKOVLV::dSetPij(iC1, TWOLIFE__XX + iStartalter + iShift[iC3] , ANNDEATH,
1. - (1. - dQxLoc)*(1. - dQyLoc));
              // now m -> f
              dQyLoc = dGetQx(SECONDORDER, RTAF, 1 /*woman*/, iYAge , iC1 + lActualYear)* psymRelQxTime
              dTemp = MARKOVLV::dSetPij(iC1, TWOLIFE__XY + iStartalter + iShift[iC3] , TWOLIFE__XY + iS
              dTemp = MARKOVLV::dSetPij(iC1, TWOLIFE__XY + iStartalter + iShift[iC3] , ANNDEATH,
1. - (1. - dQxLoc)*(1. - dQyLoc));
            }
        }

    }
  // Rente Frau
  for(iStartalter = 0; iStartalter <= MAXAGE; ++ iStartalter)
    {
      for(iC1=0; iC1<STARTTIME; ++iC1)
        {
          dQxLoc = dGetQx(SECONDORDER, RTAF, 1 /*woman*/, iStartalter + iC1 , iC1 + lActualYear)* psymR
          dTemp = MARKOVLV::dSetPij(iC1, MAN + iStartalter , MAN + iStartalter, (1. - dQxLoc));
          dTemp = MARKOVLV::dSetPij(iC1, MAN + iStartalter , ANNDEATH,  dQxLoc);
          for (iC3=0; iShift[iC3] < 100; ++iC3)
            {
              iYAge =  iStartalter + iC1 + iDXY[iC3];
              if(iYAge < 0) iYAge = 0;
              // now f -> f
              dQyLoc = dGetQx(SECONDORDER, RTAF, 1 /*woman*/, iYAge , iC1 + lActualYear)* psymRelQxTime
              dTemp = MARKOVLV::dSetPij(iC1, TWOLIFE__YY + iStartalter + iShift[iC3] , TWOLIFE__YY + iS
              dTemp = MARKOVLV::dSetPij(iC1, TWOLIFE__YY + iStartalter + iShift[iC3] , ANNDEATH,
1. - (1. - dQxLoc)*(1. - dQyLoc));
            }
        }
    }
  // 2. Belegen der Zinsen
  // -------------------------------------
  for(iC1=0; iC1<STARTTIME; ++iC1)
    {
      double dLocDisc;
      dLocDisc = psymDisc->dGetValue(iC1);
      //    printf("\ t=%d disc = %10.8f ", iC1, dLocDisc);
      for(iC2=0; iC2 < ANNNRSTATES; ++ iC2) dTemp = MARKOVLV::dSetDisc(iC1, iC2, iC2, dLocDisc);
    }
  lValid = false;
}

double          ANNMOD::dGetStatDK(void)
{
  return(dStatDK);
}

double          ANNMOD::dGetFVDK(void)
{
  return(dFVDK);
}


double        ANNMOD::dGetDK(long lTime)
{
  double dTemp=0;
```

```
  int iC1;

  if(!lValid)
    {
      vUpdateOperator();

      MARKOVLV::vSetStartTime(STARTTIME);
      MARKOVLV::vSetStopTime(STOPTIME);
      lValid = true;
    }
  // printf("\n Get GLM DK");

  for(iC1= 0; iC1 < ANNNRSTATES; ++ iC1) dTemp += MARKOVLV::dGetDK(lTime,iC1,1l);
  return(dTemp);
}

double          ANNMOD::dGetCF(long lTime)
{
  double dTemp;
  int iC1;
  if(!lValid) dTemp = dGetDK(0);
  dTemp = 0.;
  for(iC1= 0; iC1 < ANNNRSTATES; ++ iC1) dTemp += MARKOVLV::dGetCF(lTime, iC1, iC1) + MARKOVLV::dGetCF(
  return(dTemp);
}

double          ANNMOD::dGetStatDK(long lType)  // lType: 0=Stat 1=FV ohne sx
{
  if(lType == 0)       return(dStatDK);
  if(lType == 1)       return(dFVDK);

  return(0.);
}

double  ANNMOD::dSetRelativeQxForTime(long lTime, double dValue)
{
  lValid = false;
  //printf("\n Add Qx Time %5d %10lf",lTime, psymRelQxTime->dSetValue(lTime, dValue));
  return(psymRelQxTime->dSetValue(lTime, dValue));
}

double ANNMOD::dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear)
{
  long lDeltaT = lYear - lBaseYear[lOrder][lTafel][lSex];
  double dQxKorr = 1.;
  double dFxKorr = 1.;
  double dTemp;
  //if (lDeltaT > lMaxProj) lDeltaT = lMaxProj;
  //if (iQxStress == 1) dQxKorr = 1 + dQxStress;
  //if (iFxStress == 1) dFxKorr = 1 + dFxStress;
  dTemp = dQxKorr * psymQx[lOrder][lTafel][lSex]->dGetValue(lTime) * exp(dFxKorr * psymFx[lOrder][lTafe
  if(dTemp <0) dTemp =0.;
  if(dTemp >1) dTemp =1.;
  return(dTemp);
}
```

Chapter 15
make and omarkov.i

15.0.1 omarkov.i

```
/* File: example.i */
%module markovlv
%{
#define SWIG_FILE_WITH_INIT
#include "omarkov.h"
#include "annuity.h"
#include "annuity2.h"
#include "capital.h"
#include "widdow.h"
#include "annmod.h"
#include "glmod.h"
#include "vastruct_gen.h"
#include "vamod.h"
#include "tableserver.h"
%}

class MARKOVLV
{
public:
  /*  MARKOVLV(); */
  MARKOVLV(long lMaxTimesIpt, long lMaxStatesIpt, long lNrDefMomentsIpt); // Overrides Defaults
  ~MARKOVLV();
  void           vReset();   // Alles Zuruecksetzen
  void           vSetInternals(long lMaxTimes, long lMaxStates); // Diese Werte neu belegen
  void           vSetStartTime(long lTime); // Zeit an welcher Rekursion beginnt, zB 120
  void           vSetStopTime(long lTime);  // Zeit an welcher Rekursion stoppt zB 30
  void           vSetNrStates(long lNrStatesIpt); // Anzahl Zustaende des Modells
  void           vSetGetData(bool bStatus); // Falls true werden ueberschreiben die folgenden
  // 4 Funktionen keine Werte und geben die Werte nur zurueck
  // dSetPre - a_i^Pre(t)
  // dSetPost - a_{ij}^Post(t)
  // dSetPij  - p_{ij}(t)
  // dSetDisc - v_{i}(t) bzw v_{ij}(t) falls vSetInterestModel(true)
  double         dSetPre(long lTime, long lVon, long lNach, double dValue);  // lNach irrelevant
  double         dSetPost(long lTime, long lVon, long lNach, double dValue);
  double         dSetPij(long lTime, long lVon, long lNach, double dValue);
  double         dSetDisc(long lTime, long lVon, long lNach, double dValue);
  void           vSetInterestModel(bool bStocInterest);  // true heisst stochastischer Zins
  void           vSetDefaultNrMoments(long lNrMoments);  // Wenn man hoehere Momente will
  double         dGetDK(long lTime, long lState, long lMoment); // Berechnet DK's - eg V_i(t) falls
                                                                // lMoment = 1
  double         dGetCF(long lTime, long lInitState, long lTimeState); // Berechnet erwartete CF
                 // E[CF(t) x \chi_{I_t = lTimeState} | X(Stopzeit) = lInitState]
```

```
                // Wenn man den total CF will muss man also
                // summe_i dGetCF(long lTime, long lInitState, i) rechnen
  double        dGetRP(long lTime, long lState); // Berechnet Risikopraemie
  double        dGetSP(long lTime, long lState); // Berechnet Sparpraemie
  double        dGetRegP(long lTime, long lState); // Berechnet Regulaeren Zahlungsstrom
  long          lSetFolgezustand(long lStateVon, long lStateNach);
  long          lGetMaxTime();
  long          lGetNrStates();
  long          lGetStartTime();
  long          lGetStopTime();
  bool          dAddBenefits;
  void          vSetInitState(long lInitState);
  void          vGenerateTrajectory();
  long          vGetState(long lTime);
  double        dGetRandCF(long lTime);
  double        dGetRandDK(long lTime, long lMoment);
  double        dGetMeanCF(long lTime, long lState, long lNrSim);
  double        dGetMeanDK(long lTime, long lState, long lNrSim);
  void          vNewSeed(long lSeed);
  void          vResetMeanResults();
  long          lSeed;
  void vPrintTeX(FILE * psymTeXFile, bool bWithHeader, char * pcTitle, bool bAllEntries);
};

class CAPITALLV:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  CAPITALLV();
  CAPITALLV(long lMaxTimesIpt); // Overrides Defaults
  ~CAPITALLV();
  int           iSetTable(char * pcId);
  void          vSetStartTime(long lTime);
  void          vSetStopTime(long lTime);
  void          vSetSurvival(long lTime, double dValue);
  void          vSetDeath(double dValue);
  void          vSetPremium(double dValue);
  void          vSetSurvivalGen(long lTime, double dValue); // nur einzelne Werte werden
  void          vSetDeathGen(long lTime, double dValue);                // ueberschrieben statt
  void          vSetPremiumGen(long lTime, double dValue);              // allen
  double        dSetQx(long lTime, double dValue);  // lNach irrelevant
  double        dSetFx(long lTime, double dValue);
  double        dSetBaseYear(long lTime);
  double        dSetActualYear(long lTime);
  double        dSetDisc(long lTime, double dValue);
  double        dGetDK(long lTime); // Berechnet DK's
  double        dGetCF(long lTime);
  double        dGetQx(long lTime, long lYear);
  double        dSetQx2Level(long lTime, double dValue);
  double        dSetSx2(long lTime, double dValue);
  double        dSetRDR(long lTime, double dValue);
  double        dSetSurenderPenaltyInMR(long lTime, double dValue);
  double        dSetSHMarginInMR(long lTime, double dValue);
  double        dSetSolaCapitalInMR(long lTime, double dValue);
  double        dSetInvReturn(long lTime, double dValue);
  double        dGetEV(long lTime);

};

class ANNUITYLV:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  ANNUITYLV();
  ANNUITYLV(long lMaxTimesIpt, double dPre); // Overrides Defaults
  ~ANNUITYLV();
  int           iSetTable(char * pcId);
```

```
  void            vSetStartTime(long lTime);
  void            vSetStopTime(long lTime);
  void            vSetSAge(long lTime);
  void            vSetG(long lTime);
  void            vSetMaxProj(long lMaxYearImp);
  double          dSetQx(long lTime, double dValue);  // lNach irrelevant
  double          dSetFx(long lTime, double dValue);
  double          dSetSx(long lTime, double dValue);
  double          dSetBaseYear(long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);
  double          dGetDK(long lTime); // Berechnet DK's
  double          dGetCF(long lTime);
  double          dGetQx(long lTime, long lYear);
  double          dGetTqx(long lTime);
  double          dGetTpx(long lTime);
  double          dSetPre(double dValue);
  double          dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx
  void            vLeistReset();
  void            vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncrement);
  void            vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor); // eac
};

class ANNUITYLV2:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  ANNUITYLV2();
  ANNUITYLV2(long lMaxTimesIpt, double dPre); // Overrides Defaults
  ~ANNUITYLV2();
  int             iSetTable1(char * pcId);
  int             iSetTable2(char * pcId);
  void            vSetStartTime(long lTime);
  void            vSetStopTime(long lTime);
  void            vSetSAge1(long lTime);
  void            vSetSAge2(long lTime);
  double          dSetQx1(long lTime, double dValue);  // lNach irrelevant
  double          dSetFx1(long lTime, double dValue);
  double          dSetQx2(long lTime, double dValue);  // lNach irrelevant
  double          dSetFx2(long lTime, double dValue);
  double          dSetBaseYear(long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);
  double          dGetDK(long lTime, long lState); // Berechnet DK's
  double          dGetCF(long lTime);
  double          dGetQx1(long lTime, long lYear);
  double          dGetQx2(long lTime, long lYear);
  double          dSetY_Minus_X(long lYAge, long lXAge);
  double          dSetBenefit(long lState, double dValue);
  double          dSetPre(double dValue);
  void            vLeistReset();
  void            vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncrement);
  void            vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor); // eac
};

class WIDDOWLV:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
public:
  WIDDOWLV();
  WIDDOWLV(long lMaxTimesIpt); // Overrides Defaults
  ~WIDDOWLV();
  void            vSetStartTime(long lTime);
  void            vSetStopTime(long lTime);
  double          dSetQx(long lTime, double dValue);  // lNach irrelevant
  double          dSetQy(long lTime, double dValue);  // lNach irrelevant
  double          dSetFx(long lTime, double dValue);
```

```
  double          dSetFy(long lTime, double dValue);
  double          dSetHx(long lTime, double dValue);
  double          dSetYx(long lTime, double dValue);
  double          dSetBaseYear(long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);
  double          dGetDK(long lTime); // Berechnet DK's
  double          dGetCF(long lTime);
  double          dGetQx(long lTime, long lYear);
  double          dSetPre(double dValue);
  void            vLeistReset();
  void            vSetLeistLinear(long lTimeFrom, long lTimeTo, double dStartValue, double dIncrement);
  void            vSetLeistExp(long lTimeFrom, long lTimeTo, double dStartValue, double dFactor); // eac
};


class GLMOD:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
 public:
  GLMOD();
  ~GLMOD();
  double          dSetQx(long lTable, long lType, long lSex, long lTime, double dValue);  // Table 0=K,
  double          dSetFx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetSx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetBaseYear(long lTable, long lType, long lSex, long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);

  void            vStress(long lType, double dAmount); // Idee lType = 0 -> Reset  1= = ...
  void            vAddAnnuity(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem, d
  void            vAddEndowment(long lSex, long lX, long lS, double dLeist, double dPraem, double dITech
  void            vAddWiddow(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem, do
  void            vSetRKWAnnuity(long lType, double dAmount);
  void            vSetRKWEndowment(long lType, double dAmount);
  void            vUpdateOperator();

  double          dGetDK(long lTime); // Berechnet DK's
  double          dGetDKDetail(long lTime, long lState); // Berechnet DK's fuer jeden State.
  double          dGetDKTilde(long lTime); // Berechnet DK's gesehen aus Zeit Null
  double          dGetStatDK(void);
  double          dGetFVDK(void);
  double          dGetCF(long lTime);
  double          dGetCFDetail(long lTime, long lState);
  double          dGetStatDK(long lType);  // lType: 0=Stat 1=FV ohne sx
  double          dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear);
  double          dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx
  int             iReadInforce(int iP, int iL, char * strFileName);
  void            vPrintTex(char * strName);
};

class ANNMOD:MARKOVLV  // Remark to access a method of MARKOV use MARKOVLV::Method(Args)
{
 public:
  ANNMOD();
  ~ANNMOD();
  double          dSetQx(long lTable, long lType, long lSex, long lTime, double dValue);  // Table 0=K,
  double          dSetFx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetSx(long lTable, long lType, long lSex, long lTime, double dValue);
  double          dSetBaseYear(long lTable, long lType, long lSex, long lTime);
  double          dSetActualYear(long lTime);
  double          dSetDisc(long lTime, double dValue);

  // Einlebig
  void            vAddAnnuity1(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem,
  // Zinsrente
```

```
  void           vAddAnnuity0(long lX, long lS, long lNTimes, double dLeist, double dPraem, double dITe
  // Zweilebig joint life
  void           vAddAnnuity2xy(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, double
  // Zweilebig last life
  void           vAddAnnuity2xyBar(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, dou
  // Zweilebig y widdow annuity
  void           vAddAnnuity2xToy(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, doub
  // Zweilebig y widdow annuity
  void           vAddAnnuity2yTox(long lSex1, long lSex2, long lX, long lY, long lS, long lNTimes, doub


  void           vUpdateOperator();

  double         dGetDK(long lTime); // Berechnet DK's
  double         dGetStatDK(void);
  double         dGetFVDK(void);
  double         dGetCF(long lTime);
  double         dGetStatDK(long lType);  // lType: 0=Stat 1=FV ohne sx

  double         dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear);
  double         dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx
};


class VAMOD:VAINFORCE,MARKOVLV,SIMLIB
{
 public:
  VAMOD();
  ~VAMOD();
  double         dSetQx(long lTable, long lType, long lSex, long lTime, double dValue);  // Table 0=K,
  double         dSetFx(long lTable, long lType, long lSex, long lTime, double dValue);
  double         dSetSx(long lTable, long lType, long lSex, long lTime, double dValue);
  double         dSetBaseYear(long lTable, long lType, long lSex, long lTime);
  double         dSetActualYear(long lTime);
  double         dSetDisc(long lTime, double dValue);
  int            iAnalyseToken(char * pcString);

  void           vGenerateTrajectory();
  double         dGetMeanCF(long lTime, long lNrSim);
  double         dGetMeanCFAnn(long lTime, long lNrSim);
  double         dGetMeanCFPrem(long lTime, long lNrSim);
  double         dGetMeanCFMort(long lTime, long lNrSim);
  double         dGetMeanDK(long lTime, long lNrSim);
  double         dGetMeanDKAnnMort(long lTime, long lNrSim);
  double         dGetMeanDKPrem(long lTime, long lNrSim);
  double         dGetDKDetail(long lTime, long lState); // Berechnet DK's fuer jeden State.
  //  double         dGetCFDetail(long lTime, long lState);
  void           vNewSeed(long lSeed);
  void           vResetMeanResults();
  long           lSeed;

  void           vAddDeath(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem, dou
  void           vAddEndowment(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem,
  void           vAddPremium(long lSex, long lX, long lS, long lNTimes, double dLeist, double dPraem, d

  void           vUpdateOperator();

  double         dGetQx(long lOrder, long lTafel, long lSex, long lTime, long lYear);
  double         dSetRelativeQxForTime(long lTime, double dValue); // eg x_0 + time we multiply the qx
  int            iReadInforce(int iP, int iL, char * strFileName);
  void           vPrintTex(char * strName);
  VAPAR          symPara; // Parameters
  int            iSetTable(int iSex, char * pcId, double dTech);
};
```

```
class TABLESERVER
{

 public:
  TABLESERVER();
  void   vSetTable(char *pcTable);
  double dGetValue(int iAge);
  int iTableNumber();
  int iX0();
  int iXOmega();
  int iT0();
  double dITech();
  int iGender();
  char * pcAllTarifs();
};
```

15.0.2 make

```
g++ -shared -c -O3 -fPIC -DFOR_OLE  annmod.cpp
g++ -shared -c -O3 -fPIC -DFOR_OLE  annuity.cpp
g++ -shared -c -O3 -fPIC -DFOR_OLE  annuity2.cpp
g++ -shared -c -O3 -fPIC -DFOR_OLE  capital.cpp
g++ -shared -c -O3 -fPIC -DFOR_OLE  glmod.cpp
g++ -shared -c -O3 -fPIC -DFOR_OLE  omarkov.cpp
g++ -shared -c -O3 -fPIC -DFOR_OLE  widdow.cpp
swig -c++ -python omarkov.i
g++ -shared -c -O3 -fPIC -DFOR_OLE -I/usr/include/python2.6 omarkov_wrap.cxx
gcc -lstdc++ -lm -shared annmod.o annuity.o annuity2.o capital.o glmod.o omarkov.o widdow.o omarkov_wra
cp *.py* python
cp *.so python
rm *.o
rm *.py*
rm *.so
```

Chapter 16

omarkov_wrap - generated by swig

```
/* ----------------------------------------------------------------------------
 * This file was automatically generated by SWIG (http://www.swig.org).
 * Version 1.3.31
 *
 * This file is not intended to be easily readable and contains a number of
 * coding conventions designed to improve portability and efficiency. Do not make
 * changes to this file unless you know what you are doing--modify the SWIG
 * interface file instead.
 * ----------------------------------------------------------------------------- */

#define SWIGPYTHON
#define SWIG_PYTHON_DIRECTOR_NO_VTABLE

#ifdef __cplusplus
template<class T> class SwigValueWrapper {
    T *tt;
public:
    SwigValueWrapper() : tt(0) { }
    SwigValueWrapper(const SwigValueWrapper<T>& rhs) : tt(new T(*rhs.tt)) { }
    SwigValueWrapper(const T& t) : tt(new T(t)) { }
    ~SwigValueWrapper() { delete tt; }
    SwigValueWrapper& operator=(const T& t) { delete tt; tt = new T(t); return *this; }
    operator T&() const { return *tt; }
    T *operator&() { return tt; }
private:
    SwigValueWrapper& operator=(const SwigValueWrapper<T>& rhs);
};
#endif

/* ----------------------------------------------------------------------------
 *  This section contains generic SWIG labels for method/variable
 *  declarations/attributes, and other compiler dependent labels.
 * ----------------------------------------------------------------------------- */

/* template workaround for compilers that cannot correctly implement the C++ standard */
#ifndef SWIGTEMPLATEDISAMBIGUATOR
# if defined(__SUNPRO_CC)
#   if (__SUNPRO_CC <= 0x560)
```

```
#     define SWIGTEMPLATEDISAMBIGUATOR template
#   else
#     define SWIGTEMPLATEDISAMBIGUATOR
#   endif
# else
#   define SWIGTEMPLATEDISAMBIGUATOR
# endif
#endif

/* inline attribute */
#ifndef SWIGINLINE
# if defined(__cplusplus) || (defined(__GNUC__) && !defined(__STRICT_ANSI__))
#   define SWIGINLINE inline
# else
#   define SWIGINLINE
# endif
#endif

/* attribute recognised by some compilers to avoid 'unused' warnings */
#ifndef SWIGUNUSED
# if defined(__GNUC__)
#   if !(defined(__cplusplus)) || (__GNUC__ > 3 || (__GNUC__ == 3 && __GNUC_MINOR__ >= 4
#     define SWIGUNUSED __attribute__ ((__unused__))
#   else
#     define SWIGUNUSED
#   endif
# elif defined(__ICC)
#   define SWIGUNUSED __attribute__ ((__unused__))
# else
#   define SWIGUNUSED
# endif
#endif

#ifndef SWIGUNUSEDPARM
# ifdef __cplusplus
#   define SWIGUNUSEDPARM(p)
# else
#   define SWIGUNUSEDPARM(p) p SWIGUNUSED
# endif
#endif

/* internal SWIG method */
#ifndef SWIGINTERN
# define SWIGINTERN static SWIGUNUSED
#endif

/* internal inline SWIG method */
#ifndef SWIGINTERNINLINE
# define SWIGINTERNINLINE SWIGINTERN SWIGINLINE
#endif

/* exporting methods */
#if (__GNUC__ >= 4) || (__GNUC__ == 3 && __GNUC_MINOR__ >= 4)
#  ifndef GCC_HASCLASSVISIBILITY
```

```
#    define GCC_HASCLASSVISIBILITY
#  endif
#endif

#ifndef SWIGEXPORT
# if defined(_WIN32) || defined(__WIN32__) || defined(__CYGWIN__)
#   if defined(STATIC_LINKED)
#     define SWIGEXPORT
#   else
#     define SWIGEXPORT __declspec(dllexport)
#   endif
# else
#   if defined(__GNUC__) && defined(GCC_HASCLASSVISIBILITY)
#     define SWIGEXPORT __attribute__ ((visibility("default")))
#   else
#     define SWIGEXPORT
#   endif
# endif
#endif

/* calling conventions for Windows */
#ifndef SWIGSTDCALL
# if defined(_WIN32) || defined(__WIN32__) || defined(__CYGWIN__)
#   define SWIGSTDCALL __stdcall
# else
#   define SWIGSTDCALL
# endif
#endif

/* Deal with Microsoft's attempt at deprecating C standard runtime functions */
#if !defined(SWIG_NO_CRT_SECURE_NO_DEPRECATE) && defined(_MSC_VER) && !defined(_CRT_SECU
# define _CRT_SECURE_NO_DEPRECATE
#endif


/* Python.h has to appear first */
#include <Python.h>

/* -----------------------------------------------------------------------------
 * swigrun.swg
 *
 * This file contains generic CAPI SWIG runtime support for pointer
 * type checking.
 * ----------------------------------------------------------------------------- */

/* This should only be incremented when either the layout of swig_type_info changes,
   or for whatever reason, the runtime changes incompatibly */
#define SWIG_RUNTIME_VERSION "3"

/* define SWIG_TYPE_TABLE_NAME as "SWIG_TYPE_TABLE" */
#ifdef SWIG_TYPE_TABLE
# define SWIG_QUOTE_STRING(x) #x
# define SWIG_EXPAND_AND_QUOTE_STRING(x) SWIG_QUOTE_STRING(x)
# define SWIG_TYPE_TABLE_NAME SWIG_EXPAND_AND_QUOTE_STRING(SWIG_TYPE_TABLE)
```

```
#else
# define SWIG_TYPE_TABLE_NAME
#endif

/*
  You can use the SWIGRUNTIME and SWIGRUNTIMEINLINE macros for
  creating a static or dynamic library from the swig runtime code.
  In 99.9% of the cases, swig just needs to declare them as 'static'.

  But only do this if is strictly necessary, ie, if you have problems
  with your compiler or so.
*/

#ifndef SWIGRUNTIME
# define SWIGRUNTIME SWIGINTERN
#endif

#ifndef SWIGRUNTIMEINLINE
# define SWIGRUNTIMEINLINE SWIGRUNTIME SWIGINLINE
#endif

/*  Generic buffer size */
#ifndef SWIG_BUFFER_SIZE
# define SWIG_BUFFER_SIZE 1024
#endif

/* Flags for pointer conversions */
#define SWIG_POINTER_DISOWN        0x1

/* Flags for new pointer objects */
#define SWIG_POINTER_OWN           0x1


/*
   Flags/methods for returning states.

   The swig conversion methods, as ConvertPtr, return and integer
   that tells if the conversion was successful or not. And if not,
   an error code can be returned (see swigerrors.swg for the codes).

   Use the following macros/flags to set or process the returning
   states.

   In old swig versions, you usually write code as:

     if (SWIG_ConvertPtr(obj,vptr,ty.flags) != -1) {
       // success code
     } else {
       //fail code
     }

   Now you can be more explicit as:

    int res = SWIG_ConvertPtr(obj,vptr,ty.flags);
```

```
 if (SWIG_IsOK(res)) {
   // success code
 } else {
   // fail code
 }
```

that seems to be the same, but now you can also do

```
 Type *ptr;
 int res = SWIG_ConvertPtr(obj,(void **)(&ptr),ty.flags);
 if (SWIG_IsOK(res)) {
   // success code
   if (SWIG_IsNewObj(res) {
     ...
     delete *ptr;
   } else {
     ...
   }
 } else {
   // fail code
 }
```

I.e., now SWIG_ConvertPtr can return new objects and you can identify the case and take care of the deallocation. Of course that requires also to SWIG_ConvertPtr to return new result values, as

```
   int SWIG_ConvertPtr(obj, ptr,...) {
     if (<obj is ok>) {
       if (<need new object>) {
         *ptr = <ptr to new allocated object>;
         return SWIG_NEWOBJ;
       } else {
         *ptr = <ptr to old object>;
         return SWIG_OLDOBJ;
       }
     } else {
       return SWIG_BADOBJ;
     }
   }
```

Of course, returning the plain '0(success)/-1(fail)' still works, but you can be more explicit by returning SWIG_BADOBJ, SWIG_ERROR or any of the swig errors code.

Finally, if the SWIG_CASTRANK_MODE is enabled, the result code allows to return the 'cast rank', for example, if you have this

```
    int food(double)
    int fooi(int);
```

and you call

```
   food(1)   // cast rank '1'  (1 -> 1.0)
   fooi(1)   // cast rank '0'
```

```
   just use the SWIG_AddCast()/SWIG_CheckState()


 */
#define SWIG_OK                    (0)
#define SWIG_ERROR                 (-1)
#define SWIG_IsOK(r)               (r >= 0)
#define SWIG_ArgError(r)           ((r != SWIG_ERROR) ? r : SWIG_TypeError)

/* The CastRankLimit says how many bits are used for the cast rank */
#define SWIG_CASTRANKLIMIT         (1 << 8)
/* The NewMask denotes the object was created (using new/malloc) */
#define SWIG_NEWOBJMASK            (SWIG_CASTRANKLIMIT  << 1)
/* The TmpMask is for in/out typemaps that use temporal objects */
#define SWIG_TMPOBJMASK            (SWIG_NEWOBJMASK << 1)
/* Simple returning values */
#define SWIG_BADOBJ                (SWIG_ERROR)
#define SWIG_OLDOBJ                (SWIG_OK)
#define SWIG_NEWOBJ                (SWIG_OK | SWIG_NEWOBJMASK)
#define SWIG_TMPOBJ                (SWIG_OK | SWIG_TMPOBJMASK)
/* Check, add and del mask methods */
#define SWIG_AddNewMask(r)         (SWIG_IsOK(r) ? (r | SWIG_NEWOBJMASK) : r)
#define SWIG_DelNewMask(r)         (SWIG_IsOK(r) ? (r & ~SWIG_NEWOBJMASK) : r)
#define SWIG_IsNewObj(r)           (SWIG_IsOK(r) && (r & SWIG_NEWOBJMASK))
#define SWIG_AddTmpMask(r)         (SWIG_IsOK(r) ? (r | SWIG_TMPOBJMASK) : r)
#define SWIG_DelTmpMask(r)         (SWIG_IsOK(r) ? (r & ~SWIG_TMPOBJMASK) : r)
#define SWIG_IsTmpObj(r)           (SWIG_IsOK(r) && (r & SWIG_TMPOBJMASK))


/* Cast-Rank Mode */
#if defined(SWIG_CASTRANK_MODE)
#  ifndef SWIG_TypeRank
#    define SWIG_TypeRank             unsigned long
#  endif
#  ifndef SWIG_MAXCASTRANK            /* Default cast allowed */
#    define SWIG_MAXCASTRANK          (2)
#  endif
#  define SWIG_CASTRANKMASK          ((SWIG_CASTRANKLIMIT) -1)
#  define SWIG_CastRank(r)           (r & SWIG_CASTRANKMASK)
SWIGINTERNINLINE int SWIG_AddCast(int r) {
  return SWIG_IsOK(r) ? ((SWIG_CastRank(r) < SWIG_MAXCASTRANK) ? (r + 1) : SWIG_ERROR) :
}
SWIGINTERNINLINE int SWIG_CheckState(int r) {
  return SWIG_IsOK(r) ? SWIG_CastRank(r) + 1 : 0;
}
#else /* no cast-rank mode */
#  define SWIG_AddCast
#  define SWIG_CheckState(r) (SWIG_IsOK(r) ? 1 : 0)
#endif
```

```
#include <string.h>

#ifdef __cplusplus
extern "C" {
#endif

typedef void *(*swig_converter_func)(void *);
typedef struct swig_type_info *(*swig_dycast_func)(void **);

/* Structure to store inforomation on one type */
typedef struct swig_type_info {
  const char             *name;                 /* mangled name of this type */
  const char             *str;                  /* human readable name of this type */
  swig_dycast_func        dcast;                /* dynamic cast function down a hierarch
  struct swig_cast_info  *cast;                 /* linked list of types that can cast in
  void                   *clientdata;           /* language specific type data */
  int                     owndata;              /* flag if the structure owns the client
} swig_type_info;

/* Structure to store a type and conversion function used for casting */
typedef struct swig_cast_info {
  swig_type_info         *type;                 /* pointer to type that is equivalent to
  swig_converter_func     converter;            /* function to cast the void pointers */
  struct swig_cast_info  *next;                 /* pointer to next cast in linked list *
  struct swig_cast_info  *prev;                 /* pointer to the previous cast */
} swig_cast_info;

/* Structure used to store module information
 * Each module generates one structure like this, and the runtime collects
 * all of these structures and stores them in a circularly linked list.*/
typedef struct swig_module_info {
  swig_type_info         **types;               /* Array of pointers to swig_type_info s
  size_t                 size;                  /* Number of types in this module */
  struct swig_module_info *next;                /* Pointer to next element in circularly
  swig_type_info         **type_initial;        /* Array of initially generated type str
  swig_cast_info         **cast_initial;        /* Array of initially generated casting
  void                    *clientdata;          /* Language specific module data */
} swig_module_info;

/*
  Compare two type names skipping the space characters, therefore
  "char*" == "char *" and "Class<int>" == "Class<int >", etc.

  Return 0 when the two name types are equivalent, as in
  strncmp, but skipping ' '.
*/
SWIGRUNTIME int
SWIG_TypeNameComp(const char *f1, const char *l1,
                  const char *f2, const char *l2) {
  for (;(f1 != l1) && (f2 != l2); ++f1, ++f2) {
    while ((*f1 == ' ') && (f1 != l1)) ++f1;
    while ((*f2 == ' ') && (f2 != l2)) ++f2;
    if (*f1 != *f2) return (*f1 > *f2) ? 1 : -1;
  }
```

```
  return (l1 - f1) - (l2 - f2);
}

/*
  Check type equivalence in a name list like <name1>|<name2>|...
  Return 0 if not equal, 1 if equal
*/
SWIGRUNTIME int
SWIG_TypeEquiv(const char *nb, const char *tb) {
  int equiv = 0;
  const char* te = tb + strlen(tb);
  const char* ne = nb;
  while (!equiv && *ne) {
    for (nb = ne; *ne; ++ne) {
      if (*ne == '|') break;
    }
    equiv = (SWIG_TypeNameComp(nb, ne, tb, te) == 0) ? 1 : 0;
    if (*ne) ++ne;
  }
  return equiv;
}

/*
  Check type equivalence in a name list like <name1>|<name2>|...
  Return 0 if equal, -1 if nb < tb, 1 if nb > tb
*/
SWIGRUNTIME int
SWIG_TypeCompare(const char *nb, const char *tb) {
  int equiv = 0;
  const char* te = tb + strlen(tb);
  const char* ne = nb;
  while (!equiv && *ne) {
    for (nb = ne; *ne; ++ne) {
      if (*ne == '|') break;
    }
    equiv = (SWIG_TypeNameComp(nb, ne, tb, te) == 0) ? 1 : 0;
    if (*ne) ++ne;
  }
  return equiv;
}


/* think of this as a c++ template<> or a scheme macro */
#define SWIG_TypeCheck_Template(comparison, ty)         \
  if (ty) {                                             \
    swig_cast_info *iter = ty->cast;                    \
    while (iter) {                                      \
      if (comparison) {                                 \
        if (iter == ty->cast) return iter;              \
        /* Move iter to the top of the linked list */   \
        iter->prev->next = iter->next;                  \
        if (iter->next)                                 \
          iter->next->prev = iter->prev;                \
        iter->next = ty->cast;                          \
```

```
        iter->prev = 0;                                    \
        if (ty->cast) ty->cast->prev = iter;               \
        ty->cast = iter;                                   \
        return iter;                                       \
      }                                                    \
      iter = iter->next;                                   \
    }                                                      \
  }                                                        \
  return 0

/*
  Check the typename
*/
SWIGRUNTIME swig_cast_info *
SWIG_TypeCheck(const char *c, swig_type_info *ty) {
  SWIG_TypeCheck_Template(strcmp(iter->type->name, c) == 0, ty);
}

/* Same as previous function, except strcmp is replaced with a pointer comparison */
SWIGRUNTIME swig_cast_info *
SWIG_TypeCheckStruct(swig_type_info *from, swig_type_info *into) {
  SWIG_TypeCheck_Template(iter->type == from, into);
}

/*
  Cast a pointer up an inheritance hierarchy
*/
SWIGRUNTIMEINLINE void *
SWIG_TypeCast(swig_cast_info *ty, void *ptr) {
  return ((!ty) || (!ty->converter)) ? ptr : (*ty->converter)(ptr);
}

/*
   Dynamic pointer casting. Down an inheritance hierarchy
*/
SWIGRUNTIME swig_type_info *
SWIG_TypeDynamicCast(swig_type_info *ty, void **ptr) {
  swig_type_info *lastty = ty;
  if (!ty || !ty->dcast) return ty;
  while (ty && (ty->dcast)) {
    ty = (*ty->dcast)(ptr);
    if (ty) lastty = ty;
  }
  return lastty;
}

/*
  Return the name associated with this type
*/
SWIGRUNTIMEINLINE const char *
SWIG_TypeName(const swig_type_info *ty) {
  return ty->name;
}
```

```
/*
  Return the pretty name associated with this type,
  that is an unmangled type name in a form presentable to the user.
*/
SWIGRUNTIME const char *
SWIG_TypePrettyName(const swig_type_info *type) {
  /* The "str" field contains the equivalent pretty names of the
     type, separated by vertical-bar characters.  We choose
     to print the last name, as it is often (?) the most
     specific. */
  if (!type) return NULL;
  if (type->str != NULL) {
    const char *last_name = type->str;
    const char *s;
    for (s = type->str; *s; s++)
      if (*s == '|') last_name = s+1;
    return last_name;
  }
  else
    return type->name;
}

/*
   Set the clientdata field for a type
*/
SWIGRUNTIME void
SWIG_TypeClientData(swig_type_info *ti, void *clientdata) {
  swig_cast_info *cast = ti->cast;
  /* if (ti->clientdata == clientdata) return; */
  ti->clientdata = clientdata;

  while (cast) {
    if (!cast->converter) {
      swig_type_info *tc = cast->type;
      if (!tc->clientdata) {
        SWIG_TypeClientData(tc, clientdata);
      }
    }
    cast = cast->next;
  }
}
SWIGRUNTIME void
SWIG_TypeNewClientData(swig_type_info *ti, void *clientdata) {
  SWIG_TypeClientData(ti, clientdata);
  ti->owndata = 1;
}

/*
  Search for a swig_type_info structure only by mangled name
  Search is a O(log #types)

  We start searching at module start, and finish searching when start == end.
  Note: if start == end at the beginning of the function, we go all the way around
  the circular list.
```

```
*/
SWIGRUNTIME swig_type_info *
SWIG_MangledTypeQueryModule(swig_module_info *start,
                            swig_module_info *end,
                            const char *name) {
  swig_module_info *iter = start;
  do {
    if (iter->size) {
      register size_t l = 0;
      register size_t r = iter->size - 1;
      do {
        /* since l+r >= 0, we can (>> 1) instead (/ 2) */
        register size_t i = (l + r) >> 1;
        const char *iname = iter->types[i]->name;
        if (iname) {
          register int compare = strcmp(name, iname);
          if (compare == 0) {
            return iter->types[i];
          } else if (compare < 0) {
            if (i) {
              r = i - 1;
            } else {
              break;
            }
          } else if (compare > 0) {
            l = i + 1;
          }
        } else {
          break; /* should never happen */
        }
      } while (l <= r);
    }
    iter = iter->next;
  } while (iter != end);
  return 0;
}

/*
  Search for a swig_type_info structure for either a mangled name or a human readable na
  It first searches the mangled names of the types, which is a O(log #types)
  If a type is not found it then searches the human readable names, which is O(#types).

  We start searching at module start, and finish searching when start == end.
  Note: if start == end at the beginning of the function, we go all the way around
  the circular list.
*/
SWIGRUNTIME swig_type_info *
SWIG_TypeQueryModule(swig_module_info *start,
                     swig_module_info *end,
                     const char *name) {
  /* STEP 1: Search the name field using binary search */
  swig_type_info *ret = SWIG_MangledTypeQueryModule(start, end, name);
  if (ret) {
    return ret;
```

```
  } else {
    /* STEP 2: If the type hasn't been found, do a complete search
       of the str field (the human readable name) */
    swig_module_info *iter = start;
    do {
      register size_t i = 0;
      for (; i < iter->size; ++i) {
        if (iter->types[i]->str && (SWIG_TypeEquiv(iter->types[i]->str, name)))
          return iter->types[i];
      }
      iter = iter->next;
    } while (iter != end);
  }

  /* neither found a match */
  return 0;
}

/*
   Pack binary data into a string
*/
SWIGRUNTIME char *
SWIG_PackData(char *c, void *ptr, size_t sz) {
  static const char hex[17] = "0123456789abcdef";
  register const unsigned char *u = (unsigned char *) ptr;
  register const unsigned char *eu =  u + sz;
  for (; u != eu; ++u) {
    register unsigned char uu = *u;
    *(c++) = hex[(uu & 0xf0) >> 4];
    *(c++) = hex[uu & 0xf];
  }
  return c;
}

/*
   Unpack binary data from a string
*/
SWIGRUNTIME const char *
SWIG_UnpackData(const char *c, void *ptr, size_t sz) {
  register unsigned char *u = (unsigned char *) ptr;
  register const unsigned char *eu = u + sz;
  for (; u != eu; ++u) {
    register char d = *(c++);
    register unsigned char uu;
    if ((d >= '0') && (d <= '9'))
      uu = ((d - '0') << 4);
    else if ((d >= 'a') && (d <= 'f'))
      uu = ((d - ('a'-10)) << 4);
    else
      return (char *) 0;
    d = *(c++);
    if ((d >= '0') && (d <= '9'))
      uu |= (d - '0');
    else if ((d >= 'a') && (d <= 'f'))
```

```
      uu |= (d - ('a'-10));
    else
      return (char *) 0;
    *u = uu;
  }
  return c;
}

/*
   Pack 'void *' into a string buffer.
*/
SWIGRUNTIME char *
SWIG_PackVoidPtr(char *buff, void *ptr, const char *name, size_t bsz) {
  char *r = buff;
  if ((2*sizeof(void *) + 2) > bsz) return 0;
  *(r++) = '_';
  r = SWIG_PackData(r,&ptr,sizeof(void *));
  if (strlen(name) + 1 > (bsz - (r - buff))) return 0;
  strcpy(r,name);
  return buff;
}

SWIGRUNTIME const char *
SWIG_UnpackVoidPtr(const char *c, void **ptr, const char *name) {
  if (*c != '_') {
    if (strcmp(c,"NULL") == 0) {
      *ptr = (void *) 0;
      return name;
    } else {
      return 0;
    }
  }
  return SWIG_UnpackData(++c,ptr,sizeof(void *));
}

SWIGRUNTIME char *
SWIG_PackDataName(char *buff, void *ptr, size_t sz, const char *name, size_t bsz) {
  char *r = buff;
  size_t lname = (name ? strlen(name) : 0);
  if ((2*sz + 2 + lname) > bsz) return 0;
  *(r++) = '_';
  r = SWIG_PackData(r,ptr,sz);
  if (lname) {
    strncpy(r,name,lname+1);
  } else {
    *r = 0;
  }
  return buff;
}

SWIGRUNTIME const char *
SWIG_UnpackDataName(const char *c, void *ptr, size_t sz, const char *name) {
  if (*c != '_') {
    if (strcmp(c,"NULL") == 0) {
```

```
      memset(ptr,0,sz);
      return name;
    } else {
      return 0;
    }
  }
  return SWIG_UnpackData(++c,ptr,sz);
}

#ifdef __cplusplus
}
#endif

/*  Errors in SWIG */
#define  SWIG_UnknownError         -1
#define  SWIG_IOError              -2
#define  SWIG_RuntimeError         -3
#define  SWIG_IndexError           -4
#define  SWIG_TypeError            -5
#define  SWIG_DivisionByZero       -6
#define  SWIG_OverflowError        -7
#define  SWIG_SyntaxError          -8
#define  SWIG_ValueError           -9
#define  SWIG_SystemError          -10
#define  SWIG_AttributeError       -11
#define  SWIG_MemoryError          -12
#define  SWIG_NullReferenceError   -13



/* Add PyOS_snprintf for old Pythons */
#if PY_VERSION_HEX < 0x02020000
# if defined(_MSC_VER) || defined(__BORLANDC__) || defined(_WATCOM)
#  define PyOS_snprintf _snprintf
# else
#  define PyOS_snprintf snprintf
# endif
#endif

/* A crude PyString_FromFormat implementation for old Pythons */
#if PY_VERSION_HEX < 0x02020000

#ifndef SWIG_PYBUFFER_SIZE
# define SWIG_PYBUFFER_SIZE 1024
#endif

static PyObject *
PyString_FromFormat(const char *fmt, ...) {
  va_list ap;
  char buf[SWIG_PYBUFFER_SIZE * 2];
  int res;
  va_start(ap, fmt);
  res = vsnprintf(buf, sizeof(buf), fmt, ap);
```

```
  va_end(ap);
  return (res < 0 || res >= (int)sizeof(buf)) ? 0 : PyString_FromString(buf);
}
#endif

/* Add PyObject_Del for old Pythons */
#if PY_VERSION_HEX < 0x01060000
# define PyObject_Del(op) PyMem_DEL((op))
#endif
#ifndef PyObject_DEL
# define PyObject_DEL PyObject_Del
#endif

/* A crude PyExc_StopIteration exception for old Pythons */
#if PY_VERSION_HEX < 0x02020000
# ifndef PyExc_StopIteration
#  define PyExc_StopIteration PyExc_RuntimeError
# endif
# ifndef PyObject_GenericGetAttr
#  define PyObject_GenericGetAttr 0
# endif
#endif
/* Py_NotImplemented is defined in 2.1 and up. */
#if PY_VERSION_HEX < 0x02010000
# ifndef Py_NotImplemented
#  define Py_NotImplemented PyExc_RuntimeError
# endif
#endif


/* A crude PyString_AsStringAndSize implementation for old Pythons */
#if PY_VERSION_HEX < 0x02010000
# ifndef PyString_AsStringAndSize
#  define PyString_AsStringAndSize(obj, s, len) {*s = PyString_AsString(obj); *len = *s
# endif
#endif

/* PySequence_Size for old Pythons */
#if PY_VERSION_HEX < 0x02000000
# ifndef PySequence_Size
#  define PySequence_Size PySequence_Length
# endif
#endif


/* PyBool_FromLong for old Pythons */
#if PY_VERSION_HEX < 0x02030000
static
PyObject *PyBool_FromLong(long ok)
{
  PyObject *result = ok ? Py_True : Py_False;
  Py_INCREF(result);
  return result;
}
```

```
#endif

/* Py_ssize_t for old Pythons */
/* This code is as recommended by: */
/* http://www.python.org/dev/peps/pep-0353/#conversion-guidelines */
#if PY_VERSION_HEX < 0x02050000 && !defined(PY_SSIZE_T_MIN)
typedef int Py_ssize_t;
# define PY_SSIZE_T_MAX INT_MAX
# define PY_SSIZE_T_MIN INT_MIN
#endif

/* -------------------------------------------------------------------------
 * error manipulation
 * ------------------------------------------------------------------------- */

SWIGRUNTIME PyObject*
SWIG_Python_ErrorType(int code) {
  PyObject* type = 0;
  switch(code) {
  case SWIG_MemoryError:
    type = PyExc_MemoryError;
    break;
  case SWIG_IOError:
    type = PyExc_IOError;
    break;
  case SWIG_RuntimeError:
    type = PyExc_RuntimeError;
    break;
  case SWIG_IndexError:
    type = PyExc_IndexError;
    break;
  case SWIG_TypeError:
    type = PyExc_TypeError;
    break;
  case SWIG_DivisionByZero:
    type = PyExc_ZeroDivisionError;
    break;
  case SWIG_OverflowError:
    type = PyExc_OverflowError;
    break;
  case SWIG_SyntaxError:
    type = PyExc_SyntaxError;
    break;
  case SWIG_ValueError:
    type = PyExc_ValueError;
    break;
  case SWIG_SystemError:
    type = PyExc_SystemError;
    break;
  case SWIG_AttributeError:
    type = PyExc_AttributeError;
    break;
  default:
    type = PyExc_RuntimeError;
```

```
  }
  return type;
}


SWIGRUNTIME void
SWIG_Python_AddErrorMsg(const char* mesg)
{
  PyObject *type = 0;
  PyObject *value = 0;
  PyObject *traceback = 0;

  if (PyErr_Occurred()) PyErr_Fetch(&type, &value, &traceback);
  if (value) {
    PyObject *old_str = PyObject_Str(value);
    PyErr_Clear();
    Py_XINCREF(type);
    PyErr_Format(type, "%s %s", PyString_AsString(old_str), mesg);
    Py_DECREF(old_str);
    Py_DECREF(value);
  } else {
    PyErr_Format(PyExc_RuntimeError, mesg);
  }
}



#if defined(SWIG_PYTHON_NO_THREADS)
#  if defined(SWIG_PYTHON_THREADS)
#    undef SWIG_PYTHON_THREADS
#  endif
#endif
#if defined(SWIG_PYTHON_THREADS) /* Threading support is enabled */
#  if !defined(SWIG_PYTHON_USE_GIL) && !defined(SWIG_PYTHON_NO_USE_GIL)
#    if (PY_VERSION_HEX >= 0x02030000) /* For 2.3 or later, use the PyGILState calls */
#      define SWIG_PYTHON_USE_GIL
#    endif
#  endif
#  if defined(SWIG_PYTHON_USE_GIL) /* Use PyGILState threads calls */
#    ifndef SWIG_PYTHON_INITIALIZE_THREADS
#     define SWIG_PYTHON_INITIALIZE_THREADS  PyEval_InitThreads()
#    endif
#    ifdef __cplusplus /* C++ code */
       class SWIG_Python_Thread_Block {
         bool status;
         PyGILState_STATE state;
       public:
         void end() { if (status) { PyGILState_Release(state); status = false;} }
         SWIG_Python_Thread_Block() : status(true), state(PyGILState_Ensure()) {}
         ~SWIG_Python_Thread_Block() { end(); }
       };
       class SWIG_Python_Thread_Allow {
         bool status;
         PyThreadState *save;
```

```
      public:
        void end() { if (status) { PyEval_RestoreThread(save); status = false; }}
        SWIG_Python_Thread_Allow() : status(true), save(PyEval_SaveThread()) {}
        ~SWIG_Python_Thread_Allow() { end(); }
      };
#     define SWIG_PYTHON_THREAD_BEGIN_BLOCK   SWIG_Python_Thread_Block _swig_thread_blo
#     define SWIG_PYTHON_THREAD_END_BLOCK     _swig_thread_block.end()
#     define SWIG_PYTHON_THREAD_BEGIN_ALLOW   SWIG_Python_Thread_Allow _swig_thread_all
#     define SWIG_PYTHON_THREAD_END_ALLOW     _swig_thread_allow.end()
#   else /* C code */
#     define SWIG_PYTHON_THREAD_BEGIN_BLOCK   PyGILState_STATE _swig_thread_block = PyG
#     define SWIG_PYTHON_THREAD_END_BLOCK     PyGILState_Release(_swig_thread_block)
#     define SWIG_PYTHON_THREAD_BEGIN_ALLOW   PyThreadState *_swig_thread_allow = PyEva
#     define SWIG_PYTHON_THREAD_END_ALLOW     PyEval_RestoreThread(_swig_thread_allow)
#   endif
#  else /* Old thread way, not implemented, user must provide it */
#   if !defined(SWIG_PYTHON_INITIALIZE_THREADS)
#     define SWIG_PYTHON_INITIALIZE_THREADS
#   endif
#   if !defined(SWIG_PYTHON_THREAD_BEGIN_BLOCK)
#     define SWIG_PYTHON_THREAD_BEGIN_BLOCK
#   endif
#   if !defined(SWIG_PYTHON_THREAD_END_BLOCK)
#     define SWIG_PYTHON_THREAD_END_BLOCK
#   endif
#   if !defined(SWIG_PYTHON_THREAD_BEGIN_ALLOW)
#     define SWIG_PYTHON_THREAD_BEGIN_ALLOW
#   endif
#   if !defined(SWIG_PYTHON_THREAD_END_ALLOW)
#     define SWIG_PYTHON_THREAD_END_ALLOW
#   endif
#  endif
#else /* No thread support */
#  define SWIG_PYTHON_INITIALIZE_THREADS
#  define SWIG_PYTHON_THREAD_BEGIN_BLOCK
#  define SWIG_PYTHON_THREAD_END_BLOCK
#  define SWIG_PYTHON_THREAD_BEGIN_ALLOW
#  define SWIG_PYTHON_THREAD_END_ALLOW
#endif

/* -----------------------------------------------------------------------------
 * Python API portion that goes into the runtime
 * ----------------------------------------------------------------------------- */

#ifdef __cplusplus
extern "C" {
#if 0
} /* cc-mode */
#endif
#endif

/* -----------------------------------------------------------------------------
 * Constant declarations
 * ----------------------------------------------------------------------------- */
```

```
/* Constant Types */
#define SWIG_PY_POINTER 4
#define SWIG_PY_BINARY  5

/* Constant information structure */
typedef struct swig_const_info {
  int type;
  char *name;
  long lvalue;
  double dvalue;
  void   *pvalue;
  swig_type_info **ptype;
} swig_const_info;

#ifdef __cplusplus
#if 0
{ /* cc-mode */
#endif
}
#endif


/* -----------------------------------------------------------------------------
 * See the LICENSE file for information on copyright, usage and redistribution
 * of SWIG, and the README file for authors - http://www.swig.org/release.html.
 *
 * pyrun.swg
 *
 * This file contains the runtime support for Python modules
 * and includes code for managing global variables and pointer
 * type checking.
 *
 * ----------------------------------------------------------------------------- */

/* Common SWIG API */

/* for raw pointers */
#define SWIG_Python_ConvertPtr(obj, pptr, type, flags)  SWIG_Python_ConvertPtrAndOwn(obj
#define SWIG_ConvertPtr(obj, pptr, type, flags)         SWIG_Python_ConvertPtr(obj, pptr
#define SWIG_ConvertPtrAndOwn(obj,pptr,type,flags,own)  SWIG_Python_ConvertPtrAndOwn(obj
#define SWIG_NewPointerObj(ptr, type, flags)            SWIG_Python_NewPointerObj(ptr, t
#define SWIG_CheckImplicit(ty)                          SWIG_Python_CheckImplicit(ty)
#define SWIG_AcquirePtr(ptr, src)                       SWIG_Python_AcquirePtr(ptr, src)
#define swig_owntype                                    int

/* for raw packed data */
#define SWIG_ConvertPacked(obj, ptr, sz, ty)            SWIG_Python_ConvertPacked(obj, p
#define SWIG_NewPackedObj(ptr, sz, type)                SWIG_Python_NewPackedObj(ptr, sz

/* for class or struct pointers */
#define SWIG_ConvertInstance(obj, pptr, type, flags)    SWIG_ConvertPtr(obj, pptr, type,
#define SWIG_NewInstanceObj(ptr, type, flags)           SWIG_NewPointerObj(ptr, type, fl
```

```
/* for C or C++ function pointers */
#define SWIG_ConvertFunctionPtr(obj, pptr, type)        SWIG_Python_ConvertFunctionPtr(o
#define SWIG_NewFunctionPtrObj(ptr, type)               SWIG_Python_NewPointerObj(ptr, t

/* for C++ member pointers, ie, member methods */
#define SWIG_ConvertMember(obj, ptr, sz, ty)            SWIG_Python_ConvertPacked(obj, p
#define SWIG_NewMemberObj(ptr, sz, type)                SWIG_Python_NewPackedObj(ptr, sz


/* Runtime API */

#define SWIG_GetModule(clientdata)                      SWIG_Python_GetModule()
#define SWIG_SetModule(clientdata, pointer)             SWIG_Python_SetModule(pointer)
#define SWIG_NewClientData(obj)                         PySwigClientData_New(obj)

#define SWIG_SetErrorObj                                SWIG_Python_SetErrorObj
#define SWIG_SetErrorMsg                                SWIG_Python_SetErrorMsg
#define SWIG_ErrorType(code)                            SWIG_Python_ErrorType(code)
#define SWIG_Error(code, msg)                           SWIG_Python_SetErrorMsg(SWIG_Err
#define SWIG_fail                                       goto fail


/* Runtime API implementation */

/* Error manipulation */

SWIGINTERN void
SWIG_Python_SetErrorObj(PyObject *errtype, PyObject *obj) {
  SWIG_PYTHON_THREAD_BEGIN_BLOCK;
  PyErr_SetObject(errtype, obj);
  Py_DECREF(obj);
  SWIG_PYTHON_THREAD_END_BLOCK;
}

SWIGINTERN void
SWIG_Python_SetErrorMsg(PyObject *errtype, const char *msg) {
  SWIG_PYTHON_THREAD_BEGIN_BLOCK;
  PyErr_SetString(errtype, (char *) msg);
  SWIG_PYTHON_THREAD_END_BLOCK;
}

#define SWIG_Python_Raise(obj, type, desc)  SWIG_Python_SetErrorObj(SWIG_Python_Exceptio

/* Set a constant value */

SWIGINTERN void
SWIG_Python_SetConstant(PyObject *d, const char *name, PyObject *obj) {
  PyDict_SetItemString(d, (char*) name, obj);
  Py_DECREF(obj);
}

/* Append a value to the result obj */

SWIGINTERN PyObject*
```

```
SWIG_Python_AppendOutput(PyObject* result, PyObject* obj) {
#if !defined(SWIG_PYTHON_OUTPUT_TUPLE)
  if (!result) {
    result = obj;
  } else if (result == Py_None) {
    Py_DECREF(result);
    result = obj;
  } else {
    if (!PyList_Check(result)) {
      PyObject *o2 = result;
      result = PyList_New(1);
      PyList_SetItem(result, 0, o2);
    }
    PyList_Append(result,obj);
    Py_DECREF(obj);
  }
  return result;
#else
  PyObject*   o2;
  PyObject*   o3;
  if (!result) {
    result = obj;
  } else if (result == Py_None) {
    Py_DECREF(result);
    result = obj;
  } else {
    if (!PyTuple_Check(result)) {
      o2 = result;
      result = PyTuple_New(1);
      PyTuple_SET_ITEM(result, 0, o2);
    }
    o3 = PyTuple_New(1);
    PyTuple_SET_ITEM(o3, 0, obj);
    o2 = result;
    result = PySequence_Concat(o2, o3);
    Py_DECREF(o2);
    Py_DECREF(o3);
  }
  return result;
#endif
}

/* Unpack the argument tuple */

SWIGINTERN int
SWIG_Python_UnpackTuple(PyObject *args, const char *name, int min, int max, PyObject **o
{
  if (!args) {
    if (!min && !max) {
      return 1;
    } else {
      PyErr_Format(PyExc_TypeError, "%s expected %s%d arguments, got none",
                   name, (min == max ? "" : "at least "), min);
      return 0;
```

```
    }
  }
  if (!PyTuple_Check(args)) {
    PyErr_SetString(PyExc_SystemError, "UnpackTuple() argument list is not a tuple");
    return 0;
  } else {
    register int l = PyTuple_GET_SIZE(args);
    if (l < min) {
      PyErr_Format(PyExc_TypeError, "%s expected %s%d arguments, got %d",
                   name, (min == max ? "" : "at least "), min, l);
      return 0;
    } else if (l > max) {
      PyErr_Format(PyExc_TypeError, "%s expected %s%d arguments, got %d",
                   name, (min == max ? "" : "at most "), max, l);
      return 0;
    } else {
      register int i;
      for (i = 0; i < l; ++i) {
        objs[i] = PyTuple_GET_ITEM(args, i);
      }
      for (; l < max; ++l) {
        objs[l] = 0;
      }
      return i + 1;
    }
  }
}

/* A functor is a function object with one single object argument */
#if PY_VERSION_HEX >= 0x02020000
#define SWIG_Python_CallFunctor(functor, obj)           PyObject_CallFunctionObjArgs(fur
#else
#define SWIG_Python_CallFunctor(functor, obj)           PyObject_CallFunction(functor, '
#endif

/*
  Helper for static pointer initialization for both C and C++ code, for example
  static PyObject *SWIG_STATIC_POINTER(MyVar) = NewSomething(...);
*/
#ifdef __cplusplus
#define SWIG_STATIC_POINTER(var)  var
#else
#define SWIG_STATIC_POINTER(var)  var = 0; if (!var) var
#endif

/* -----------------------------------------------------------------------------
 * Pointer declarations
 * ----------------------------------------------------------------------------- */

/* Flags for new pointer objects */
#define SWIG_POINTER_NOSHADOW       (SWIG_POINTER_OWN      << 1)
#define SWIG_POINTER_NEW            (SWIG_POINTER_NOSHADOW | SWIG_POINTER_OWN)

#define SWIG_POINTER_IMPLICIT_CONV  (SWIG_POINTER_DISOWN   << 1)
```

```
#ifdef __cplusplus
extern "C" {
#if 0
} /* cc-mode */
#endif
#endif

/*  How to access Py_None */
#if defined(_WIN32) || defined(__WIN32__) || defined(__CYGWIN__)
#  ifndef SWIG_PYTHON_NO_BUILD_NONE
#    ifndef SWIG_PYTHON_BUILD_NONE
#      define SWIG_PYTHON_BUILD_NONE
#    endif
#  endif
#endif

#ifdef SWIG_PYTHON_BUILD_NONE
#  ifdef Py_None
#   undef Py_None
#   define Py_None SWIG_Py_None()
#  endif
SWIGRUNTIMEINLINE PyObject *
_SWIG_Py_None(void)
{
  PyObject *none = Py_BuildValue((char*)"");
  Py_DECREF(none);
  return none;
}
SWIGRUNTIME PyObject *
SWIG_Py_None(void)
{
  static PyObject *SWIG_STATIC_POINTER(none) = _SWIG_Py_None();
  return none;
}
#endif

/* The python void return value */

SWIGRUNTIMEINLINE PyObject *
SWIG_Py_Void(void)
{
  PyObject *none = Py_None;
  Py_INCREF(none);
  return none;
}

/* PySwigClientData */

typedef struct {
  PyObject *klass;
  PyObject *newraw;
  PyObject *newargs;
  PyObject *destroy;
```

```
  int delargs;
  int implicitconv;
} PySwigClientData;

SWIGRUNTIMEINLINE int
SWIG_Python_CheckImplicit(swig_type_info *ty)
{
  PySwigClientData *data = (PySwigClientData *)ty->clientdata;
  return data ? data->implicitconv : 0;
}

SWIGRUNTIMEINLINE PyObject *
SWIG_Python_ExceptionType(swig_type_info *desc) {
  PySwigClientData *data = desc ? (PySwigClientData *) desc->clientdata : 0;
  PyObject *klass = data ? data->klass : 0;
  return (klass ? klass : PyExc_RuntimeError);
}


SWIGRUNTIME PySwigClientData *
PySwigClientData_New(PyObject* obj)
{
  if (!obj) {
    return 0;
  } else {
    PySwigClientData *data = (PySwigClientData *)malloc(sizeof(PySwigClientData));
    /* the klass element */
    data->klass = obj;
    Py_INCREF(data->klass);
    /* the newraw method and newargs arguments used to create a new raw instance */
    if (PyClass_Check(obj)) {
      data->newraw = 0;
      data->newargs = obj;
      Py_INCREF(obj);
    } else {
#if (PY_VERSION_HEX < 0x02020000)
      data->newraw = 0;
#else
      data->newraw = PyObject_GetAttrString(data->klass, (char *)"__new__");
#endif
      if (data->newraw) {
        Py_INCREF(data->newraw);
        data->newargs = PyTuple_New(1);
        PyTuple_SetItem(data->newargs, 0, obj);
      } else {
        data->newargs = obj;
      }
      Py_INCREF(data->newargs);
    }
    /* the destroy method, aka as the C++ delete method */
    data->destroy = PyObject_GetAttrString(data->klass, (char *)"__swig_destroy__");
    if (PyErr_Occurred()) {
      PyErr_Clear();
      data->destroy = 0;
```

```
    }
    if (data->destroy) {
      int flags;
      Py_INCREF(data->destroy);
      flags = PyCFunction_GET_FLAGS(data->destroy);
#ifdef METH_O
      data->delargs = !(flags & (METH_O));
#else
      data->delargs = 0;
#endif
    } else {
      data->delargs = 0;
    }
    data->implicitconv = 0;
    return data;
  }
}

SWIGRUNTIME void
PySwigClientData_Del(PySwigClientData* data)
{
  Py_XDECREF(data->newraw);
  Py_XDECREF(data->newargs);
  Py_XDECREF(data->destroy);
}

/* =============== PySwigObject =====================*/

typedef struct {
  PyObject_HEAD
  void *ptr;
  swig_type_info *ty;
  int own;
  PyObject *next;
} PySwigObject;

SWIGRUNTIME PyObject *
PySwigObject_long(PySwigObject *v)
{
  return PyLong_FromVoidPtr(v->ptr);
}

SWIGRUNTIME PyObject *
PySwigObject_format(const char* fmt, PySwigObject *v)
{
  PyObject *res = NULL;
  PyObject *args = PyTuple_New(1);
  if (args) {
    if (PyTuple_SetItem(args, 0, PySwigObject_long(v)) == 0) {
      PyObject *ofmt = PyString_FromString(fmt);
      if (ofmt) {
        res = PyString_Format(ofmt,args);
        Py_DECREF(ofmt);
      }
```

```
      Py_DECREF(args);
    }
  }
  return res;
}

SWIGRUNTIME PyObject *
PySwigObject_oct(PySwigObject *v)
{
  return PySwigObject_format("%o",v);
}

SWIGRUNTIME PyObject *
PySwigObject_hex(PySwigObject *v)
{
  return PySwigObject_format("%x",v);
}

SWIGRUNTIME PyObject *
#ifdef METH_NOARGS
PySwigObject_repr(PySwigObject *v)
#else
PySwigObject_repr(PySwigObject *v, PyObject *args)
#endif
{
  const char *name = SWIG_TypePrettyName(v->ty);
  PyObject *hex = PySwigObject_hex(v);
  PyObject *repr = PyString_FromFormat("<Swig Object of type '%s' at 0x%s>", name, PyStr
  Py_DECREF(hex);
  if (v->next) {
#ifdef METH_NOARGS
    PyObject *nrep = PySwigObject_repr((PySwigObject *)v->next);
#else
    PyObject *nrep = PySwigObject_repr((PySwigObject *)v->next, args);
#endif
    PyString_ConcatAndDel(&repr,nrep);
  }
  return repr;
}

SWIGRUNTIME int
PySwigObject_print(PySwigObject *v, FILE *fp, int SWIGUNUSEDPARM(flags))
{
#ifdef METH_NOARGS
  PyObject *repr = PySwigObject_repr(v);
#else
  PyObject *repr = PySwigObject_repr(v, NULL);
#endif
  if (repr) {
    fputs(PyString_AsString(repr), fp);
    Py_DECREF(repr);
    return 0;
  } else {
    return 1;
```

```
  }
}

SWIGRUNTIME PyObject *
PySwigObject_str(PySwigObject *v)
{
  char result[SWIG_BUFFER_SIZE];
  return SWIG_PackVoidPtr(result, v->ptr, v->ty->name, sizeof(result)) ?
    PyString_FromString(result) : 0;
}

SWIGRUNTIME int
PySwigObject_compare(PySwigObject *v, PySwigObject *w)
{
  void *i = v->ptr;
  void *j = w->ptr;
  return (i < j) ? -1 : ((i > j) ? 1 : 0);
}

SWIGRUNTIME PyTypeObject* _PySwigObject_type(void);

SWIGRUNTIME PyTypeObject*
PySwigObject_type(void) {
  static PyTypeObject *SWIG_STATIC_POINTER(type) = _PySwigObject_type();
  return type;
}

SWIGRUNTIMEINLINE int
PySwigObject_Check(PyObject *op) {
  return ((op)->ob_type == PySwigObject_type())
    || (strcmp((op)->ob_type->tp_name,"PySwigObject") == 0);
}

SWIGRUNTIME PyObject *
PySwigObject_New(void *ptr, swig_type_info *ty, int own);

SWIGRUNTIME void
PySwigObject_dealloc(PyObject *v)
{
  PySwigObject *sobj = (PySwigObject *) v;
  PyObject *next = sobj->next;
  if (sobj->own) {
    swig_type_info *ty = sobj->ty;
    PySwigClientData *data = ty ? (PySwigClientData *) ty->clientdata : 0;
    PyObject *destroy = data ? data->destroy : 0;
    if (destroy) {
      /* destroy is always a VARARGS method */
      PyObject *res;
      if (data->delargs) {
        /* we need to create a temporal object to carry the destroy operation */
        PyObject *tmp = PySwigObject_New(sobj->ptr, ty, 0);
        res = SWIG_Python_CallFunctor(destroy, tmp);
        Py_DECREF(tmp);
      } else {
```

```
        PyCFunction meth = PyCFunction_GET_FUNCTION(destroy);
        PyObject *mself = PyCFunction_GET_SELF(destroy);
        res = ((*meth)(mself, v));
      }
      Py_XDECREF(res);
    } else {
      const char *name = SWIG_TypePrettyName(ty);
#if !defined(SWIG_PYTHON_SILENT_MEMLEAK)
      printf("swig/python detected a memory leak of type '%s', no destructor found.\n",
#endif
    }
  }
  Py_XDECREF(next);
  PyObject_DEL(v);
}

SWIGRUNTIME PyObject*
PySwigObject_append(PyObject* v, PyObject* next)
{
  PySwigObject *sobj = (PySwigObject *) v;
#ifndef METH_O
  PyObject *tmp = 0;
  if (!PyArg_ParseTuple(next,(char *)"O:append", &tmp)) return NULL;
  next = tmp;
#endif
  if (!PySwigObject_Check(next)) {
    return NULL;
  }
  sobj->next = next;
  Py_INCREF(next);
  return SWIG_Py_Void();
}

SWIGRUNTIME PyObject*
#ifdef METH_NOARGS
PySwigObject_next(PyObject* v)
#else
PySwigObject_next(PyObject* v, PyObject *SWIGUNUSEDPARM(args))
#endif
{
  PySwigObject *sobj = (PySwigObject *) v;
  if (sobj->next) {
    Py_INCREF(sobj->next);
    return sobj->next;
  } else {
    return SWIG_Py_Void();
  }
}

SWIGINTERN PyObject*
#ifdef METH_NOARGS
PySwigObject_disown(PyObject *v)
#else
PySwigObject_disown(PyObject* v, PyObject *SWIGUNUSEDPARM(args))
```

```
#endif
{
  PySwigObject *sobj = (PySwigObject *)v;
  sobj->own = 0;
  return SWIG_Py_Void();
}

SWIGINTERN PyObject*
#ifdef METH_NOARGS
PySwigObject_acquire(PyObject *v)
#else
PySwigObject_acquire(PyObject* v, PyObject *SWIGUNUSEDPARM(args))
#endif
{
  PySwigObject *sobj = (PySwigObject *)v;
  sobj->own = SWIG_POINTER_OWN;
  return SWIG_Py_Void();
}

SWIGINTERN PyObject*
PySwigObject_own(PyObject *v, PyObject *args)
{
  PyObject *val = 0;
#if (PY_VERSION_HEX < 0x02020000)
  if (!PyArg_ParseTuple(args,(char *)"|O:own",&val))
#else
  if (!PyArg_UnpackTuple(args, (char *)"own", 0, 1, &val))
#endif
    {
      return NULL;
    }
  else
    {
      PySwigObject *sobj = (PySwigObject *)v;
      PyObject *obj = PyBool_FromLong(sobj->own);
      if (val) {
#ifdef METH_NOARGS
        if (PyObject_IsTrue(val)) {
          PySwigObject_acquire(v);
        } else {
          PySwigObject_disown(v);
        }
#else
        if (PyObject_IsTrue(val)) {
          PySwigObject_acquire(v,args);
        } else {
          PySwigObject_disown(v,args);
        }
#endif
      }
      return obj;
    }
}
```

```
#ifdef METH_O
static PyMethodDef
swigobject_methods[] = {
  {(char *)"disown",  (PyCFunction)PySwigObject_disown,  METH_NOARGS,  (char *)"releases
  {(char *)"acquire", (PyCFunction)PySwigObject_acquire, METH_NOARGS,  (char *)"aquires
  {(char *)"own",     (PyCFunction)PySwigObject_own,     METH_VARARGS, (char *)"returns/
  {(char *)"append",  (PyCFunction)PySwigObject_append,  METH_O,       (char *)"appends
  {(char *)"next",    (PyCFunction)PySwigObject_next,    METH_NOARGS,  (char *)"returns
  {(char *)"__repr__",(PyCFunction)PySwigObject_repr,    METH_NOARGS,  (char *)"returns
  {0, 0, 0, 0}
};
#else
static PyMethodDef
swigobject_methods[] = {
  {(char *)"disown",  (PyCFunction)PySwigObject_disown,  METH_VARARGS,  (char *)"release
  {(char *)"acquire", (PyCFunction)PySwigObject_acquire, METH_VARARGS,  (char *)"aquires
  {(char *)"own",     (PyCFunction)PySwigObject_own,     METH_VARARGS,  (char *)"returns
  {(char *)"append",  (PyCFunction)PySwigObject_append,  METH_VARARGS,  (char *)"appends
  {(char *)"next",    (PyCFunction)PySwigObject_next,    METH_VARARGS,  (char *)"returns
  {(char *)"__repr__",(PyCFunction)PySwigObject_repr,   METH_VARARGS,  (char *)"returns
  {0, 0, 0, 0}
};
#endif

#if PY_VERSION_HEX < 0x02020000
SWIGINTERN PyObject *
PySwigObject_getattr(PySwigObject *sobj,char *name)
{
  return Py_FindMethod(swigobject_methods, (PyObject *)sobj, name);
}
#endif

SWIGRUNTIME PyTypeObject*
_PySwigObject_type(void) {
  static char swigobject_doc[] = "Swig object carries a C/C++ instance pointer";

  static PyNumberMethods PySwigObject_as_number = {
    (binaryfunc)0, /*nb_add*/
    (binaryfunc)0, /*nb_subtract*/
    (binaryfunc)0, /*nb_multiply*/
    (binaryfunc)0, /*nb_divide*/
    (binaryfunc)0, /*nb_remainder*/
    (binaryfunc)0, /*nb_divmod*/
    (ternaryfunc)0,/*nb_power*/
    (unaryfunc)0,  /*nb_negative*/
    (unaryfunc)0,  /*nb_positive*/
    (unaryfunc)0,  /*nb_absolute*/
    (inquiry)0,    /*nb_nonzero*/
    0,             /*nb_invert*/
    0,             /*nb_lshift*/
    0,             /*nb_rshift*/
    0,             /*nb_and*/
    0,             /*nb_xor*/
    0,             /*nb_or*/
```

```
    (coercion)0,    /*nb_coerce*/
    (unaryfunc)PySwigObject_long, /*nb_int*/
    (unaryfunc)PySwigObject_long, /*nb_long*/
    (unaryfunc)0,                 /*nb_float*/
    (unaryfunc)PySwigObject_oct,  /*nb_oct*/
    (unaryfunc)PySwigObject_hex,  /*nb_hex*/
#if PY_VERSION_HEX >= 0x02020000
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 /* nb_inplace_add -> nb_inplace_true_divide */
#elif PY_VERSION_HEX >= 0x02000000
    0,0,0,0,0,0,0,0,0,0,0 /* nb_inplace_add -> nb_inplace_or */
#endif
  };

  static PyTypeObject pyswigobject_type;
  static int type_init = 0;
  if (!type_init) {
    const PyTypeObject tmp
      = {
        PyObject_HEAD_INIT(NULL)
        0,                                  /* ob_size */
        (char *)"PySwigObject",             /* tp_name */
        sizeof(PySwigObject),               /* tp_basicsize */
        0,                                  /* tp_itemsize */
        (destructor)PySwigObject_dealloc,   /* tp_dealloc */
        (printfunc)PySwigObject_print,      /* tp_print */
#if PY_VERSION_HEX < 0x02020000
        (getattrfunc)PySwigObject_getattr,  /* tp_getattr */
#else
        (getattrfunc)0,                     /* tp_getattr */
#endif
        (setattrfunc)0,                     /* tp_setattr */
        (cmpfunc)PySwigObject_compare,      /* tp_compare */
        (reprfunc)PySwigObject_repr,        /* tp_repr */
        &PySwigObject_as_number,            /* tp_as_number */
        0,                                  /* tp_as_sequence */
        0,                                  /* tp_as_mapping */
        (hashfunc)0,                        /* tp_hash */
        (ternaryfunc)0,                     /* tp_call */
        (reprfunc)PySwigObject_str,         /* tp_str */
        PyObject_GenericGetAttr,            /* tp_getattro */
        0,                                  /* tp_setattro */
        0,                                  /* tp_as_buffer */
        Py_TPFLAGS_DEFAULT,                 /* tp_flags */
        swigobject_doc,                     /* tp_doc */
        0,                                  /* tp_traverse */
        0,                                  /* tp_clear */
        0,                                  /* tp_richcompare */
        0,                                  /* tp_weaklistoffset */
#if PY_VERSION_HEX >= 0x02020000
        0,                                  /* tp_iter */
        0,                                  /* tp_iternext */
        swigobject_methods,                 /* tp_methods */
        0,                                  /* tp_members */
        0,                                  /* tp_getset */
```

```
        0,                                      /* tp_base */
        0,                                      /* tp_dict */
        0,                                      /* tp_descr_get */
        0,                                      /* tp_descr_set */
        0,                                      /* tp_dictoffset */
        0,                                      /* tp_init */
        0,                                      /* tp_alloc */
        0,                                      /* tp_new */
        0,                                      /* tp_free */
        0,                                      /* tp_is_gc */
        0,                                      /* tp_bases */
        0,                                      /* tp_mro */
        0,                                      /* tp_cache */
        0,                                      /* tp_subclasses */
        0,                                      /* tp_weaklist */
#endif
#if PY_VERSION_HEX >= 0x02030000
        0,                                      /* tp_del */
#endif
#ifdef COUNT_ALLOCS
        0,0,0,0                                 /* tp_alloc -> tp_next */
#endif
      };
    pyswigobject_type = tmp;
    pyswigobject_type.ob_type = &PyType_Type;
    type_init = 1;
  }
  return &pyswigobject_type;
}

SWIGRUNTIME PyObject *
PySwigObject_New(void *ptr, swig_type_info *ty, int own)
{
  PySwigObject *sobj = PyObject_NEW(PySwigObject, PySwigObject_type());
  if (sobj) {
    sobj->ptr  = ptr;
    sobj->ty   = ty;
    sobj->own  = own;
    sobj->next = 0;
  }
  return (PyObject *)sobj;
}

/* -----------------------------------------------------------------------------
 * Implements a simple Swig Packed type, and use it instead of string
 * ----------------------------------------------------------------------------- */

typedef struct {
  PyObject_HEAD
  void *pack;
  swig_type_info *ty;
  size_t size;
} PySwigPacked;
```

```
SWIGRUNTIME int
PySwigPacked_print(PySwigPacked *v, FILE *fp, int SWIGUNUSEDPARM(flags))
{
  char result[SWIG_BUFFER_SIZE];
  fputs("<Swig Packed ", fp);
  if (SWIG_PackDataName(result, v->pack, v->size, 0, sizeof(result))) {
    fputs("at ", fp);
    fputs(result, fp);
  }
  fputs(v->ty->name,fp);
  fputs(">", fp);
  return 0;
}

SWIGRUNTIME PyObject *
PySwigPacked_repr(PySwigPacked *v)
{
  char result[SWIG_BUFFER_SIZE];
  if (SWIG_PackDataName(result, v->pack, v->size, 0, sizeof(result))) {
    return PyString_FromFormat("<Swig Packed at %s%s>", result, v->ty->name);
  } else {
    return PyString_FromFormat("<Swig Packed %s>", v->ty->name);
  }
}

SWIGRUNTIME PyObject *
PySwigPacked_str(PySwigPacked *v)
{
  char result[SWIG_BUFFER_SIZE];
  if (SWIG_PackDataName(result, v->pack, v->size, 0, sizeof(result))){
    return PyString_FromFormat("%s%s", result, v->ty->name);
  } else {
    return PyString_FromString(v->ty->name);
  }
}

SWIGRUNTIME int
PySwigPacked_compare(PySwigPacked *v, PySwigPacked *w)
{
  size_t i = v->size;
  size_t j = w->size;
  int s = (i < j) ? -1 : ((i > j) ? 1 : 0);
  return s ? s : strncmp((char *)v->pack, (char *)w->pack, 2*v->size);
}

SWIGRUNTIME PyTypeObject* _PySwigPacked_type(void);

SWIGRUNTIME PyTypeObject*
PySwigPacked_type(void) {
  static PyTypeObject *SWIG_STATIC_POINTER(type) = _PySwigPacked_type();
  return type;
}

SWIGRUNTIMEINLINE int
```

```
PySwigPacked_Check(PyObject *op) {
  return ((op)->ob_type == _PySwigPacked_type())
    || (strcmp((op)->ob_type->tp_name,"PySwigPacked") == 0);
}

SWIGRUNTIME void
PySwigPacked_dealloc(PyObject *v)
{
  if (PySwigPacked_Check(v)) {
    PySwigPacked *sobj = (PySwigPacked *) v;
    free(sobj->pack);
  }
  PyObject_DEL(v);
}

SWIGRUNTIME PyTypeObject*
_PySwigPacked_type(void) {
  static char swigpacked_doc[] = "Swig object carries a C/C++ instance pointer";
  static PyTypeObject pyswigpacked_type;
  static int type_init = 0;
  if (!type_init) {
    const PyTypeObject tmp
      = {
        PyObject_HEAD_INIT(NULL)
        0,                                  /* ob_size */
        (char *)"PySwigPacked",             /* tp_name */
        sizeof(PySwigPacked),               /* tp_basicsize */
        0,                                  /* tp_itemsize */
        (destructor)PySwigPacked_dealloc,   /* tp_dealloc */
        (printfunc)PySwigPacked_print,      /* tp_print */
        (getattrfunc)0,                     /* tp_getattr */
        (setattrfunc)0,                     /* tp_setattr */
        (cmpfunc)PySwigPacked_compare,      /* tp_compare */
        (reprfunc)PySwigPacked_repr,        /* tp_repr */
        0,                                  /* tp_as_number */
        0,                                  /* tp_as_sequence */
        0,                                  /* tp_as_mapping */
        (hashfunc)0,                        /* tp_hash */
        (ternaryfunc)0,                     /* tp_call */
        (reprfunc)PySwigPacked_str,         /* tp_str */
        PyObject_GenericGetAttr,            /* tp_getattro */
        0,                                  /* tp_setattro */
        0,                                  /* tp_as_buffer */
        Py_TPFLAGS_DEFAULT,                 /* tp_flags */
        swigpacked_doc,                     /* tp_doc */
        0,                                  /* tp_traverse */
        0,                                  /* tp_clear */
        0,                                  /* tp_richcompare */
        0,                                  /* tp_weaklistoffset */
#if PY_VERSION_HEX >= 0x02020000
        0,                                  /* tp_iter */
        0,                                  /* tp_iternext */
        0,                                  /* tp_methods */
        0,                                  /* tp_members */
```

```
        0,                                      /* tp_getset */
        0,                                      /* tp_base */
        0,                                      /* tp_dict */
        0,                                      /* tp_descr_get */
        0,                                      /* tp_descr_set */
        0,                                      /* tp_dictoffset */
        0,                                      /* tp_init */
        0,                                      /* tp_alloc */
        0,                                      /* tp_new */
        0,                                      /* tp_free */
        0,                                      /* tp_is_gc */
        0,                                      /* tp_bases */
        0,                                      /* tp_mro */
        0,                                      /* tp_cache */
        0,                                      /* tp_subclasses */
        0,                                      /* tp_weaklist */
#endif
#if PY_VERSION_HEX >= 0x02030000
        0,                                      /* tp_del */
#endif
#ifdef COUNT_ALLOCS
        0,0,0,0                                 /* tp_alloc -> tp_next */
#endif
      };
    pyswigpacked_type = tmp;
    pyswigpacked_type.ob_type = &PyType_Type;
    type_init = 1;
  }
  return &pyswigpacked_type;
}

SWIGRUNTIME PyObject *
PySwigPacked_New(void *ptr, size_t size, swig_type_info *ty)
{
  PySwigPacked *sobj = PyObject_NEW(PySwigPacked, PySwigPacked_type());
  if (sobj) {
    void *pack = malloc(size);
    if (pack) {
      memcpy(pack, ptr, size);
      sobj->pack = pack;
      sobj->ty   = ty;
      sobj->size = size;
    } else {
      PyObject_DEL((PyObject *) sobj);
      sobj = 0;
    }
  }
  return (PyObject *) sobj;
}

SWIGRUNTIME swig_type_info *
PySwigPacked_UnpackData(PyObject *obj, void *ptr, size_t size)
{
  if (PySwigPacked_Check(obj)) {
```

```
    PySwigPacked *sobj = (PySwigPacked *)obj;
    if (sobj->size != size) return 0;
    memcpy(ptr, sobj->pack, size);
    return sobj->ty;
  } else {
    return 0;
  }
}

/* -----------------------------------------------------------------------------
 * pointers/data manipulation
 * ----------------------------------------------------------------------------- */

SWIGRUNTIMEINLINE PyObject *
_SWIG_This(void)
{
  return PyString_FromString("this");
}

SWIGRUNTIME PyObject *
SWIG_This(void)
{
  static PyObject *SWIG_STATIC_POINTER(swig_this) = _SWIG_This();
  return swig_this;
}

/* #define SWIG_PYTHON_SLOW_GETSET_THIS */

SWIGRUNTIME PySwigObject *
SWIG_Python_GetSwigThis(PyObject *pyobj)
{
  if (PySwigObject_Check(pyobj)) {
    return (PySwigObject *) pyobj;
  } else {
    PyObject *obj = 0;
#if (!defined(SWIG_PYTHON_SLOW_GETSET_THIS) && (PY_VERSION_HEX >= 0x02030000))
    if (PyInstance_Check(pyobj)) {
      obj = _PyInstance_Lookup(pyobj, SWIG_This());
    } else {
      PyObject **dictptr = _PyObject_GetDictPtr(pyobj);
      if (dictptr != NULL) {
        PyObject *dict = *dictptr;
        obj = dict ? PyDict_GetItem(dict, SWIG_This()) : 0;
      } else {
#ifdef PyWeakref_CheckProxy
        if (PyWeakref_CheckProxy(pyobj)) {
          PyObject *wobj = PyWeakref_GET_OBJECT(pyobj);
          return wobj ? SWIG_Python_GetSwigThis(wobj) : 0;
        }
#endif
        obj = PyObject_GetAttr(pyobj,SWIG_This());
        if (obj) {
          Py_DECREF(obj);
        } else {
```

```
          if (PyErr_Occurred()) PyErr_Clear();
          return 0;
        }
      }
    }
#else
    obj = PyObject_GetAttr(pyobj,SWIG_This());
    if (obj) {
      Py_DECREF(obj);
    } else {
      if (PyErr_Occurred()) PyErr_Clear();
      return 0;
    }
#endif
    if (obj && !PySwigObject_Check(obj)) {
      /* a PyObject is called 'this', try to get the 'real this'
         PySwigObject from it */
      return SWIG_Python_GetSwigThis(obj);
    }
    return (PySwigObject *)obj;
  }
}

/* Acquire a pointer value */

SWIGRUNTIME int
SWIG_Python_AcquirePtr(PyObject *obj, int own) {
  if (own) {
    PySwigObject *sobj = SWIG_Python_GetSwigThis(obj);
    if (sobj) {
      int oldown = sobj->own;
      sobj->own = own;
      return oldown;
    }
  }
  return 0;
}

/* Convert a pointer value */

SWIGRUNTIME int
SWIG_Python_ConvertPtrAndOwn(PyObject *obj, void **ptr, swig_type_info *ty, int flags, i
  if (!obj) return SWIG_ERROR;
  if (obj == Py_None) {
    if (ptr) *ptr = 0;
    return SWIG_OK;
  } else {
    PySwigObject *sobj = SWIG_Python_GetSwigThis(obj);
    while (sobj) {
      void *vptr = sobj->ptr;
      if (ty) {
        swig_type_info *to = sobj->ty;
        if (to == ty) {
          /* no type cast needed */
```

```
        if (ptr) *ptr = vptr;
        break;
      } else {
        swig_cast_info *tc = SWIG_TypeCheck(to->name,ty);
        if (!tc) {
          sobj = (PySwigObject *)sobj->next;
        } else {
          if (ptr) *ptr = SWIG_TypeCast(tc,vptr);
          break;
        }
      }
    } else {
      if (ptr) *ptr = vptr;
      break;
    }
  }
  if (sobj) {
    if (own) *own = sobj->own;
    if (flags & SWIG_POINTER_DISOWN) {
      sobj->own = 0;
    }
    return SWIG_OK;
  } else {
    int res = SWIG_ERROR;
    if (flags & SWIG_POINTER_IMPLICIT_CONV) {
      PySwigClientData *data = ty ? (PySwigClientData *) ty->clientdata : 0;
      if (data && !data->implicitconv) {
        PyObject *klass = data->klass;
        if (klass) {
          PyObject *impconv;
          data->implicitconv = 1; /* avoid recursion and call 'explicit' constructors*
          impconv = SWIG_Python_CallFunctor(klass, obj);
          data->implicitconv = 0;
          if (PyErr_Occurred()) {
            PyErr_Clear();
            impconv = 0;
          }
          if (impconv) {
            PySwigObject *iobj = SWIG_Python_GetSwigThis(impconv);
            if (iobj) {
              void *vptr;
              res = SWIG_Python_ConvertPtrAndOwn((PyObject*)iobj, &vptr, ty, 0, 0);
              if (SWIG_IsOK(res)) {
                if (ptr) {
                  *ptr = vptr;
                  /* transfer the ownership to 'ptr' */
                  iobj->own = 0;
                  res = SWIG_AddCast(res);
                  res = SWIG_AddNewMask(res);
                } else {
                  res = SWIG_AddCast(res);
                }
              }
            }
```

```
              Py_DECREF(impconv);
            }
          }
        }
      }
      return res;
    }
  }
}

/* Convert a function ptr value */

SWIGRUNTIME int
SWIG_Python_ConvertFunctionPtr(PyObject *obj, void **ptr, swig_type_info *ty) {
  if (!PyCFunction_Check(obj)) {
    return SWIG_ConvertPtr(obj, ptr, ty, 0);
  } else {
    void *vptr = 0;

    /* here we get the method pointer for callbacks */
    const char *doc = (((PyCFunctionObject *)obj) -> m_ml -> ml_doc);
    const char *desc = doc ? strstr(doc, "swig_ptr: ") : 0;
    if (desc) {
      desc = ty ? SWIG_UnpackVoidPtr(desc + 10, &vptr, ty->name) : 0;
      if (!desc) return SWIG_ERROR;
    }
    if (ty) {
      swig_cast_info *tc = SWIG_TypeCheck(desc,ty);
      if (!tc) return SWIG_ERROR;
      *ptr = SWIG_TypeCast(tc,vptr);
    } else {
      *ptr = vptr;
    }
    return SWIG_OK;
  }
}

/* Convert a packed value value */

SWIGRUNTIME int
SWIG_Python_ConvertPacked(PyObject *obj, void *ptr, size_t sz, swig_type_info *ty) {
  swig_type_info *to = PySwigPacked_UnpackData(obj, ptr, sz);
  if (!to) return SWIG_ERROR;
  if (ty) {
    if (to != ty) {
      /* check type cast? */
      swig_cast_info *tc = SWIG_TypeCheck(to->name,ty);
      if (!tc) return SWIG_ERROR;
    }
  }
  return SWIG_OK;
}

/* ----------------------------------------------------------------------------
```

```
 * Create a new pointer object
 * ----------------------------------------------------------------------------- */

/*
  Create a new instance object, whitout calling __init__, and set the
  'this' attribute.
*/

SWIGRUNTIME PyObject*
SWIG_Python_NewShadowInstance(PySwigClientData *data, PyObject *swig_this)
{
#if (PY_VERSION_HEX >= 0x02020000)
  PyObject *inst = 0;
  PyObject *newraw = data->newraw;
  if (newraw) {
    inst = PyObject_Call(newraw, data->newargs, NULL);
    if (inst) {
#if !defined(SWIG_PYTHON_SLOW_GETSET_THIS)
      PyObject **dictptr = _PyObject_GetDictPtr(inst);
      if (dictptr != NULL) {
        PyObject *dict = *dictptr;
        if (dict == NULL) {
          dict = PyDict_New();
          *dictptr = dict;
          PyDict_SetItem(dict, SWIG_This(), swig_this);
        }
      }
#else
      PyObject *key = SWIG_This();
      PyObject_SetAttr(inst, key, swig_this);
#endif
    }
  } else {
    PyObject *dict = PyDict_New();
    PyDict_SetItem(dict, SWIG_This(), swig_this);
    inst = PyInstance_NewRaw(data->newargs, dict);
    Py_DECREF(dict);
  }
  return inst;
#else
#if (PY_VERSION_HEX >= 0x02010000)
  PyObject *inst;
  PyObject *dict = PyDict_New();
  PyDict_SetItem(dict, SWIG_This(), swig_this);
  inst = PyInstance_NewRaw(data->newargs, dict);
  Py_DECREF(dict);
  return (PyObject *) inst;
#else
  PyInstanceObject *inst = PyObject_NEW(PyInstanceObject, &PyInstance_Type);
  if (inst == NULL) {
    return NULL;
  }
  inst->in_class = (PyClassObject *)data->newargs;
  Py_INCREF(inst->in_class);
```

```
  inst->in_dict = PyDict_New();
  if (inst->in_dict == NULL) {
    Py_DECREF(inst);
    return NULL;
  }
#ifdef Py_TPFLAGS_HAVE_WEAKREFS
  inst->in_weakreflist = NULL;
#endif
#ifdef Py_TPFLAGS_GC
  PyObject_GC_Init(inst);
#endif
  PyDict_SetItem(inst->in_dict, SWIG_This(), swig_this);
  return (PyObject *) inst;
#endif
#endif
}

SWIGRUNTIME void
SWIG_Python_SetSwigThis(PyObject *inst, PyObject *swig_this)
{
 PyObject *dict;
#if (PY_VERSION_HEX >= 0x02020000) && !defined(SWIG_PYTHON_SLOW_GETSET_THIS)
 PyObject **dictptr = _PyObject_GetDictPtr(inst);
 if (dictptr != NULL) {
   dict = *dictptr;
   if (dict == NULL) {
     dict = PyDict_New();
     *dictptr = dict;
   }
   PyDict_SetItem(dict, SWIG_This(), swig_this);
   return;
 }
#endif
 dict = PyObject_GetAttrString(inst, (char*)"__dict__");
 PyDict_SetItem(dict, SWIG_This(), swig_this);
 Py_DECREF(dict);
}


SWIGINTERN PyObject *
SWIG_Python_InitShadowInstance(PyObject *args) {
  PyObject *obj[2];
  if (!SWIG_Python_UnpackTuple(args,(char*)"swiginit", 2, 2, obj)) {
    return NULL;
  } else {
    PySwigObject *sthis = SWIG_Python_GetSwigThis(obj[0]);
    if (sthis) {
      PySwigObject_append((PyObject*) sthis, obj[1]);
    } else {
      SWIG_Python_SetSwigThis(obj[0], obj[1]);
    }
    return SWIG_Py_Void();
  }
}
```

```
/* Create a new pointer object */

SWIGRUNTIME PyObject *
SWIG_Python_NewPointerObj(void *ptr, swig_type_info *type, int flags) {
  if (!ptr) {
    return SWIG_Py_Void();
  } else {
    int own = (flags & SWIG_POINTER_OWN) ? SWIG_POINTER_OWN : 0;
    PyObject *robj = PySwigObject_New(ptr, type, own);
    PySwigClientData *clientdata = type ? (PySwigClientData *)(type->clientdata) : 0;
    if (clientdata && !(flags & SWIG_POINTER_NOSHADOW)) {
      PyObject *inst = SWIG_Python_NewShadowInstance(clientdata, robj);
      if (inst) {
        Py_DECREF(robj);
        robj = inst;
      }
    }
    return robj;
  }
}

/* Create a new packed object */

SWIGRUNTIMEINLINE PyObject *
SWIG_Python_NewPackedObj(void *ptr, size_t sz, swig_type_info *type) {
  return ptr ? PySwigPacked_New((void *) ptr, sz, type) : SWIG_Py_Void();
}

/* -----------------------------------------------------------------------------*
 *  Get type list
 * -----------------------------------------------------------------------------*/

#ifdef SWIG_LINK_RUNTIME
void *SWIG_ReturnGlobalTypeList(void *);
#endif

SWIGRUNTIME swig_module_info *
SWIG_Python_GetModule(void) {
  static void *type_pointer = (void *)0;
  /* first check if module already created */
  if (!type_pointer) {
#ifdef SWIG_LINK_RUNTIME
    type_pointer = SWIG_ReturnGlobalTypeList((void *)0);
#else
    type_pointer = PyCObject_Import((char*)"swig_runtime_data" SWIG_RUNTIME_VERSION,
                                    (char*)"type_pointer" SWIG_TYPE_TABLE_NAME);
    if (PyErr_Occurred()) {
      PyErr_Clear();
      type_pointer = (void *)0;
    }
#endif
  }
  return (swig_module_info *) type_pointer;
```

```
}

#if PY_MAJOR_VERSION < 2
/* PyModule_AddObject function was introduced in Python 2.0.  The following function
   is copied out of Python/modsupport.c in python version 2.3.4 */
SWIGINTERN int
PyModule_AddObject(PyObject *m, char *name, PyObject *o)
{
  PyObject *dict;
  if (!PyModule_Check(m)) {
    PyErr_SetString(PyExc_TypeError,
                    "PyModule_AddObject() needs module as first arg");
    return SWIG_ERROR;
  }
  if (!o) {
    PyErr_SetString(PyExc_TypeError,
                    "PyModule_AddObject() needs non-NULL value");
    return SWIG_ERROR;
  }

  dict = PyModule_GetDict(m);
  if (dict == NULL) {
    /* Internal error -- modules must have a dict! */
    PyErr_Format(PyExc_SystemError, "module '%s' has no __dict__",
                 PyModule_GetName(m));
    return SWIG_ERROR;
  }
  if (PyDict_SetItemString(dict, name, o))
    return SWIG_ERROR;
  Py_DECREF(o);
  return SWIG_OK;
}
#endif

SWIGRUNTIME void
SWIG_Python_DestroyModule(void *vptr)
{
  swig_module_info *swig_module = (swig_module_info *) vptr;
  swig_type_info **types = swig_module->types;
  size_t i;
  for (i =0; i < swig_module->size; ++i) {
    swig_type_info *ty = types[i];
    if (ty->owndata) {
      PySwigClientData *data = (PySwigClientData *) ty->clientdata;
      if (data) PySwigClientData_Del(data);
    }
  }
  Py_DECREF(SWIG_This());
}

SWIGRUNTIME void
SWIG_Python_SetModule(swig_module_info *swig_module) {
  static PyMethodDef swig_empty_runtime_method_table[] = { {NULL, NULL, 0, NULL} };/* Se
```

```
  PyObject *module = Py_InitModule((char*)"swig_runtime_data" SWIG_RUNTIME_VERSION,
                                   swig_empty_runtime_method_table);
  PyObject *pointer = PyCObject_FromVoidPtr((void *) swig_module, SWIG_Python_DestroyMod
  if (pointer && module) {
    PyModule_AddObject(module, (char*)"type_pointer" SWIG_TYPE_TABLE_NAME, pointer);
  } else {
    Py_XDECREF(pointer);
  }
}

/* The python cached type query */
SWIGRUNTIME PyObject *
SWIG_Python_TypeCache(void) {
  static PyObject *SWIG_STATIC_POINTER(cache) = PyDict_New();
  return cache;
}

SWIGRUNTIME swig_type_info *
SWIG_Python_TypeQuery(const char *type)
{
  PyObject *cache = SWIG_Python_TypeCache();
  PyObject *key = PyString_FromString(type);
  PyObject *obj = PyDict_GetItem(cache, key);
  swig_type_info *descriptor;
  if (obj) {
    descriptor = (swig_type_info *) PyCObject_AsVoidPtr(obj);
  } else {
    swig_module_info *swig_module = SWIG_Python_GetModule();
    descriptor = SWIG_TypeQueryModule(swig_module, swig_module, type);
    if (descriptor) {
      obj = PyCObject_FromVoidPtr(descriptor, NULL);
      PyDict_SetItem(cache, key, obj);
      Py_DECREF(obj);
    }
  }
  Py_DECREF(key);
  return descriptor;
}

/*
   For backward compatibility only
*/
#define SWIG_POINTER_EXCEPTION  0
#define SWIG_arg_fail(arg)      SWIG_Python_ArgFail(arg)
#define SWIG_MustGetPtr(p, type, argnum, flags)  SWIG_Python_MustGetPtr(p, type, argnum,

SWIGRUNTIME int
SWIG_Python_AddErrMesg(const char* mesg, int infront)
{
  if (PyErr_Occurred()) {
    PyObject *type = 0;
    PyObject *value = 0;
    PyObject *traceback = 0;
    PyErr_Fetch(&type, &value, &traceback);
```

```
    if (value) {
      PyObject *old_str = PyObject_Str(value);
      Py_XINCREF(type);
      PyErr_Clear();
      if (infront) {
        PyErr_Format(type, "%s %s", mesg, PyString_AsString(old_str));
      } else {
        PyErr_Format(type, "%s %s", PyString_AsString(old_str), mesg);
      }
      Py_DECREF(old_str);
    }
    return 1;
  } else {
    return 0;
  }
}

SWIGRUNTIME int
SWIG_Python_ArgFail(int argnum)
{
  if (PyErr_Occurred()) {
    /* add information about failing argument */
    char mesg[256];
    PyOS_snprintf(mesg, sizeof(mesg), "argument number %d:", argnum);
    return SWIG_Python_AddErrMesg(mesg, 1);
  } else {
    return 0;
  }
}

SWIGRUNTIMEINLINE const char *
PySwigObject_GetDesc(PyObject *self)
{
  PySwigObject *v = (PySwigObject *)self;
  swig_type_info *ty = v ? v->ty : 0;
  return ty ? ty->str : (char*)"";
}

SWIGRUNTIME void
SWIG_Python_TypeError(const char *type, PyObject *obj)
{
  if (type) {
#if defined(SWIG_COBJECT_TYPES)
    if (obj && PySwigObject_Check(obj)) {
      const char *otype = (const char *) PySwigObject_GetDesc(obj);
      if (otype) {
        PyErr_Format(PyExc_TypeError, "a '%s' is expected, 'PySwigObject(%s)' is receive
                     type, otype);
        return;
      }
    } else
#endif
    {
      const char *otype = (obj ? obj->ob_type->tp_name : 0);
```

```
      if (otype) {
        PyObject *str = PyObject_Str(obj);
        const char *cstr = str ? PyString_AsString(str) : 0;
        if (cstr) {
          PyErr_Format(PyExc_TypeError, "a '%s' is expected, '%s(%s)' is received",
                       type, otype, cstr);
        } else {
          PyErr_Format(PyExc_TypeError, "a '%s' is expected, '%s' is received",
                       type, otype);
        }
        Py_XDECREF(str);
        return;
      }
    }
    PyErr_Format(PyExc_TypeError, "a '%s' is expected", type);
  } else {
    PyErr_Format(PyExc_TypeError, "unexpected type is received");
  }
}


/* Convert a pointer value, signal an exception on a type mismatch */
SWIGRUNTIME void *
SWIG_Python_MustGetPtr(PyObject *obj, swig_type_info *ty, int argnum, int flags) {
  void *result;
  if (SWIG_Python_ConvertPtr(obj, &result, ty, flags) == -1) {
    PyErr_Clear();
    if (flags & SWIG_POINTER_EXCEPTION) {
      SWIG_Python_TypeError(SWIG_TypePrettyName(ty), obj);
      SWIG_Python_ArgFail(argnum);
    }
  }
  return result;
}


#ifdef __cplusplus
#if 0
{ /* cc-mode */
#endif
}
#endif



#define SWIG_exception_fail(code, msg) do { SWIG_Error(code, msg); SWIG_fail; } while(0)

#define SWIG_contract_assert(expr, msg) if (!(expr)) { SWIG_Error(SWIG_RuntimeError, msg



/* -------- TYPES TABLE (BEGIN) -------- */

#define SWIGTYPE_p_ANNMOD swig_types[0]
```

```
#define SWIGTYPE_p_ANNUITYLV swig_types[1]
#define SWIGTYPE_p_ANNUITYLV2 swig_types[2]
#define SWIGTYPE_p_CAPITALLV swig_types[3]
#define SWIGTYPE_p_FILE swig_types[4]
#define SWIGTYPE_p_GLMOD swig_types[5]
#define SWIGTYPE_p_MARKOVLV swig_types[6]
#define SWIGTYPE_p_TABLESERVER swig_types[7]
#define SWIGTYPE_p_VAMOD swig_types[8]
#define SWIGTYPE_p_VAPAR swig_types[9]
#define SWIGTYPE_p_WIDDOWLV swig_types[10]
#define SWIGTYPE_p_char swig_types[11]
static swig_type_info *swig_types[13];
static swig_module_info swig_module = {swig_types, 12, 0, 0, 0, 0};
#define SWIG_TypeQuery(name) SWIG_TypeQueryModule(&swig_module, &swig_module, name)
#define SWIG_MangledTypeQuery(name) SWIG_MangledTypeQueryModule(&swig_module, &swig_modu

/* -------- TYPES TABLE (END) -------- */

#if (PY_VERSION_HEX <= 0x02000000)
# if !defined(SWIG_PYTHON_CLASSIC)
#  error "This python version requires swig to be run with the '-classic' option"
# endif
#endif

/*-----------------------------------------------
              @(target):= _markovlv.so
  ------------------------------------------------*/
#define SWIG_init    init_markovlv

#define SWIG_name    "_markovlv"

#define SWIGVERSION 0x010331
#define SWIG_VERSION SWIGVERSION


#define SWIG_as_voidptr(a) const_cast< void * >(static_cast< const void * >(a))
#define SWIG_as_voidptrptr(a) ((void)SWIG_as_voidptr(*a),reinterpret_cast< void** >(a))


#include <stdexcept>


namespace swig {
  class PyObject_ptr {
  protected:
    PyObject *_obj;

  public:
    PyObject_ptr() :_obj(0)
    {
    }

    PyObject_ptr(const PyObject_ptr& item) : _obj(item._obj)
    {
```

```
      Py_XINCREF(_obj);
    }

    PyObject_ptr(PyObject *obj, bool initial_ref = true) :_obj(obj)
    {
      if (initial_ref) Py_XINCREF(_obj);
    }

    PyObject_ptr & operator=(const PyObject_ptr& item)
    {
      Py_XINCREF(item._obj);
      Py_XDECREF(_obj);
      _obj = item._obj;
      return *this;
    }

    ~PyObject_ptr()
    {
      Py_XDECREF(_obj);
    }

    operator PyObject *() const
    {
      return _obj;
    }

    PyObject *operator->() const
    {
      return _obj;
    }
  };
}


namespace swig {
  struct PyObject_var : PyObject_ptr {
    PyObject_var(PyObject* obj = 0) : PyObject_ptr(obj, false) { }

    PyObject_var & operator = (PyObject* obj)
    {
      Py_XDECREF(_obj);
      _obj = obj;
      return *this;
    }
  };
}


#define SWIG_FILE_WITH_INIT
#include "omarkov.h"
#include "annuity.h"
#include "annuity2.h"
#include "capital.h"
#include "widdow.h"
```

```
#include "annmod.h"
#include "glmod.h"
#include "vastruct_gen.h"
#include "vamod.h"
#include "tableserver.h"


SWIGINTERN int
SWIG_AsVal_double (PyObject *obj, double *val)
{
  int res = SWIG_TypeError;
  if (PyFloat_Check(obj)) {
    if (val) *val = PyFloat_AsDouble(obj);
    return SWIG_OK;
  } else if (PyInt_Check(obj)) {
    if (val) *val = PyInt_AsLong(obj);
    return SWIG_OK;
  } else if (PyLong_Check(obj)) {
    double v = PyLong_AsDouble(obj);
    if (!PyErr_Occurred()) {
      if (val) *val = v;
      return SWIG_OK;
    } else {
      PyErr_Clear();
    }
  }
#ifdef SWIG_PYTHON_CAST_MODE
  {
    int dispatch = 0;
    double d = PyFloat_AsDouble(obj);
    if (!PyErr_Occurred()) {
      if (val) *val = d;
      return SWIG_AddCast(SWIG_OK);
    } else {
      PyErr_Clear();
    }
    if (!dispatch) {
      long v = PyLong_AsLong(obj);
      if (!PyErr_Occurred()) {
        if (val) *val = v;
        return SWIG_AddCast(SWIG_AddCast(SWIG_OK));
      } else {
        PyErr_Clear();
      }
    }
  }
#endif
  return res;
}


#include <float.h>
```

```
#include <math.h>


SWIGINTERNINLINE int
SWIG_CanCastAsInteger(double *d, double min, double max) {
  double x = *d;
  if ((min <= x && x <= max)) {
   double fx = floor(x);
   double cx = ceil(x);
   double rd =  ((x - fx) < 0.5) ? fx : cx; /* simple rint */
   if ((errno == EDOM) || (errno == ERANGE)) {
     errno = 0;
   } else {
     double summ, reps, diff;
     if (rd < x) {
       diff = x - rd;
     } else if (rd > x) {
       diff = rd - x;
     } else {
       return 1;
     }
     summ = rd + x;
     reps = diff/summ;
     if (reps < 8*DBL_EPSILON) {
       *d = rd;
       return 1;
     }
   }
  }
  return 0;
}


SWIGINTERN int
SWIG_AsVal_long (PyObject *obj, long* val)
{
  if (PyInt_Check(obj)) {
    if (val) *val = PyInt_AsLong(obj);
    return SWIG_OK;
  } else if (PyLong_Check(obj)) {
    long v = PyLong_AsLong(obj);
    if (!PyErr_Occurred()) {
      if (val) *val = v;
      return SWIG_OK;
    } else {
      PyErr_Clear();
    }
  }
#ifdef SWIG_PYTHON_CAST_MODE
  {
    int dispatch = 0;
    long v = PyInt_AsLong(obj);
    if (!PyErr_Occurred()) {
      if (val) *val = v;
```

```
      return SWIG_AddCast(SWIG_OK);
    } else {
      PyErr_Clear();
    }
    if (!dispatch) {
      double d;
      int res = SWIG_AddCast(SWIG_AsVal_double (obj,&d));
      if (SWIG_IsOK(res) && SWIG_CanCastAsInteger(&d, LONG_MIN, LONG_MAX)) {
        if (val) *val = (long)(d);
        return res;
      }
    }
  }
#endif
  return SWIG_TypeError;
}


SWIGINTERN int
SWIG_AsVal_bool (PyObject *obj, bool *val)
{
  if (obj == Py_True) {
    if (val) *val = true;
    return SWIG_OK;
  } else if (obj == Py_False) {
    if (val) *val = false;
    return SWIG_OK;
  } else {
    long v = 0;
    int res = SWIG_AddCast(SWIG_AsVal_long (obj, val ? &v : 0));
    if (SWIG_IsOK(res) && val) *val = v ? true : false;
    return res;
  }
}


  #define SWIG_From_double   PyFloat_FromDouble


  #define SWIG_From_long   PyInt_FromLong


SWIGINTERNINLINE PyObject*
  SWIG_From_bool  (bool value)
{
  return PyBool_FromLong(value ? 1 : 0);
}


SWIGINTERN swig_type_info*
SWIG_pchar_descriptor(void)
{
  static int init = 0;
  static swig_type_info* info = 0;
```

```
  if (!init) {
    info = SWIG_TypeQuery("_p_char");
    init = 1;
  }
  return info;
}


SWIGINTERN int
SWIG_AsCharPtrAndSize(PyObject *obj, char** cptr, size_t* psize, int *alloc)
{
  if (PyString_Check(obj)) {
    char *cstr; Py_ssize_t len;
    PyString_AsStringAndSize(obj, &cstr, &len);
    if (cptr)  {
      if (alloc) {
        /*
           In python the user should not be able to modify the inner
           string representation. To warranty that, if you define
           SWIG_PYTHON_SAFE_CSTRINGS, a new/copy of the python string
           buffer is always returned.

           The default behavior is just to return the pointer value,
           so, be careful.
        */
#if defined(SWIG_PYTHON_SAFE_CSTRINGS)
        if (*alloc != SWIG_OLDOBJ)
#else
        if (*alloc == SWIG_NEWOBJ)
#endif
          {
            *cptr = reinterpret_cast< char* >(memcpy((new char[len + 1]), cstr, sizeof(c
            *alloc = SWIG_NEWOBJ;
          }
        else {
          *cptr = cstr;
          *alloc = SWIG_OLDOBJ;
        }
      } else {
        *cptr = PyString_AsString(obj);
      }
    }
    if (psize) *psize = len + 1;
    return SWIG_OK;
  } else {
    swig_type_info* pchar_descriptor = SWIG_pchar_descriptor();
    if (pchar_descriptor) {
      void* vptr = 0;
      if (SWIG_ConvertPtr(obj, &vptr, pchar_descriptor, 0) == SWIG_OK) {
        if (cptr) *cptr = (char *) vptr;
        if (psize) *psize = vptr ? (strlen((char *)vptr) + 1) : 0;
        if (alloc) *alloc = SWIG_OLDOBJ;
        return SWIG_OK;
      }
```

```
    }
  }
  return SWIG_TypeError;
}



SWIGINTERNINLINE PyObject *
SWIG_From_int  (int value)
{
  return SWIG_From_long  (value);
}


#include <limits.h>
#ifndef LLONG_MIN
# define LLONG_MIN      LONG_LONG_MIN
#endif
#ifndef LLONG_MAX
# define LLONG_MAX      LONG_LONG_MAX
#endif
#ifndef ULLONG_MAX
# define ULLONG_MAX     ULONG_LONG_MAX
#endif


SWIGINTERN int
SWIG_AsVal_int (PyObject * obj, int *val)
{
  long v;
  int res = SWIG_AsVal_long (obj, &v);
  if (SWIG_IsOK(res)) {
    if ((v < INT_MIN || v > INT_MAX)) {
      return SWIG_OverflowError;
    } else {
      if (val) *val = static_cast< int >(v);
    }
  }
  return res;
}


SWIGINTERNINLINE PyObject *
SWIG_FromCharPtrAndSize(const char* carray, size_t size)
{
  if (carray) {
    if (size > INT_MAX) {
      swig_type_info* pchar_descriptor = SWIG_pchar_descriptor();
      return pchar_descriptor ?
        SWIG_NewPointerObj(const_cast< char * >(carray), pchar_descriptor, 0) : SWIG_Py_
    } else {
      return PyString_FromStringAndSize(carray, static_cast< int >(size));
```

```
    }
  } else {
    return SWIG_Py_Void();
  }
}


SWIGINTERNINLINE PyObject *
SWIG_FromCharPtr(const char *cptr)
{
  return SWIG_FromCharPtrAndSize(cptr, (cptr ? strlen(cptr) : 0));
}

#ifdef __cplusplus
extern "C" {
#endif
SWIGINTERN PyObject *_wrap_new_MARKOVLV(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  long arg1 ;
  long arg2 ;
  long arg3 ;
  MARKOVLV *result = 0 ;
  long val1 ;
  int ecode1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:new_MARKOVLV",&obj0,&obj1,&obj2)) SWIG_fail;
  ecode1 = SWIG_AsVal_long(obj0, &val1);
  if (!SWIG_IsOK(ecode1)) {
    SWIG_exception_fail(SWIG_ArgError(ecode1), "in method '" "new_MARKOVLV" "', argument
  }
  arg1 = static_cast< long >(val1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "new_MARKOVLV" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "new_MARKOVLV" "', argument
  }
  arg3 = static_cast< long >(val3);
  result = (MARKOVLV *)new MARKOVLV(arg1,arg2,arg3);
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_MARKOVLV, SWIG_POIN
0 );
  return resultobj;
fail:
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_delete_MARKOVLV(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_MARKOVLV",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_MARKOVLV" "', argumen
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vReset(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_vReset",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vReset" "', argumen
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  (arg1)->vReset();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetInternals(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:MARKOVLV_vSetInternals",&obj0,&obj1,&obj2)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetInternals" "',
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetInternals" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_vSetInternals" "'
  }
  arg3 = static_cast< long >(val3);
  (arg1)->vSetInternals(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetStartTime(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetStartTime",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetStartTime" "',
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetStartTime" "'
  }
  arg2 = static_cast< long >(val2);
```

```
  (arg1)->vSetStartTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetStopTime(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetStopTime",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetStopTime" "', a
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetStopTime" "',
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStopTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetNrStates(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetNrStates",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetNrStates" "', a
  }
```

```
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetNrStates" "',
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetNrStates(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetGetData(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  bool arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  bool val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetGetData",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetGetData" "', ar
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_bool(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetGetData" "',
  }
  arg2 = static_cast< bool >(val2);
  (arg1)->vSetGetData(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dSetPre(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double arg5 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:MARKOVLV_dSetPre",&obj0,&obj1,&obj2,&obj3,&o
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dSetPre" "', argume
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dSetPre" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dSetPre" "', argu
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dSetPre" "', argu
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "MARKOVLV_dSetPre" "', argu
  }
  arg5 = static_cast< double >(val5);
  result = (double)(arg1)->dSetPre(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dSetPost(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
```

```
  double arg5 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:MARKOVLV_dSetPost",&obj0,&obj1,&obj2,&obj3,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dSetPost" "', argum
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dSetPost" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dSetPost" "', arg
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dSetPost" "', arg
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "MARKOVLV_dSetPost" "', arg
  }
  arg5 = static_cast< double >(val5);
  result = (double)(arg1)->dSetPost(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dSetPij(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
```

```
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double arg5 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:MARKOVLV_dSetPij",&obj0,&obj1,&obj2,&obj3,&o
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dSetPij" "', argume
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dSetPij" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dSetPij" "', argu
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dSetPij" "', argu
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "MARKOVLV_dSetPij" "', argu
  }
  arg5 = static_cast< double >(val5);
  result = (double)(arg1)->dSetPij(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_MARKOVLV_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double arg5 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:MARKOVLV_dSetDisc",&obj0,&obj1,&obj2,&obj3,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dSetDisc" "', argum
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dSetDisc" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dSetDisc" "', arg
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dSetDisc" "', arg
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "MARKOVLV_dSetDisc" "', arg
  }
  arg5 = static_cast< double >(val5);
  result = (double)(arg1)->dSetDisc(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
```

```
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetInterestModel(PyObject *SWIGUNUSEDPARM(self), Py
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  bool arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  bool val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetInterestModel",&obj0,&obj1)) SWIG_
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetInterestModel"
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_bool(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetInterestModel
  }
  arg2 = static_cast< bool >(val2);
  (arg1)->vSetInterestModel(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetDefaultNrMoments(PyObject *SWIGUNUSEDPARM(self),
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetDefaultNrMoments",&obj0,&obj1)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetDefaultNrMoment
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
```

```
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetDefaultNrMome
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetDefaultNrMoments(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:MARKOVLV_dGetDK",&obj0,&obj1,&obj2,&obj3)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetDK" "', argumer
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetDK" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetDK" "', argum
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dGetDK" "', argum
  }
  arg4 = static_cast< long >(val4);
  result = (double)(arg1)->dGetDK(arg2,arg3,arg4);
```

```
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:MARKOVLV_dGetCF",&obj0,&obj1,&obj2,&obj3)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetCF" "', argumer
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetCF" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetCF" "', argum
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dGetCF" "', argum
  }
  arg4 = static_cast< long >(val4);
  result = (double)(arg1)->dGetCF(arg2,arg3,arg4);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_MARKOVLV_dGetRP(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:MARKOVLV_dGetRP",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetRP" "', argumen
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetRP" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetRP" "', argum
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetRP(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetSP(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
```

```
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:MARKOVLV_dGetSP",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetSP" "', argumer
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetSP" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetSP" "', argum
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetSP(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetRegP(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:MARKOVLV_dGetRegP",&obj0,&obj1,&obj2)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetRegP" "', argum
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetRegP" "', arg
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetRegP" "', arg
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetRegP(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_lSetFolgezustand(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:MARKOVLV_lSetFolgezustand",&obj0,&obj1,&obj2))
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lSetFolgezustand" '
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_lSetFolgezustand'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_lSetFolgezustand'
  }
  arg3 = static_cast< long >(val3);
  result = (long)(arg1)->lSetFolgezustand(arg2,arg3);
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_MARKOVLV_lGetMaxTime(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_lGetMaxTime",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lGetMaxTime" "', ar
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  result = (long)(arg1)->lGetMaxTime();
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_lGetNrStates(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_lGetNrStates",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lGetNrStates" "', a
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  result = (long)(arg1)->lGetNrStates();
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_lGetStartTime(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_lGetStartTime",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
```

```
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lGetStartTime" "',
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  result = (long)(arg1)->lGetStartTime();
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_lGetStopTime(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_lGetStopTime",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lGetStopTime" "', a
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  result = (long)(arg1)->lGetStopTime();
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dAddBenefits_set(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  bool arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  bool val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_dAddBenefits_set",&obj0,&obj1)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dAddBenefits_set" '
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_bool(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dAddBenefits_set'
```

```
  }
  arg2 = static_cast< bool >(val2);
  if (arg1) (arg1)->dAddBenefits = arg2;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dAddBenefits_get(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  bool result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_dAddBenefits_get",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dAddBenefits_get" '
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  result = (bool) ((arg1)->dAddBenefits);
  resultobj = SWIG_From_bool(static_cast< bool >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vSetInitState(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vSetInitState",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vSetInitState" "',
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vSetInitState" "'
  }
```

```
  arg2 = static_cast< long >(val2);
  (arg1)->vSetInitState(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vGenerateTrajectory(PyObject *SWIGUNUSEDPARM(self),
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_vGenerateTrajectory",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vGenerateTrajectory
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  (arg1)->vGenerateTrajectory();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vGetState(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vGetState",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vGetState" "', argu
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vGetState" "', ar
  }
  arg2 = static_cast< long >(val2);
  result = (long)(arg1)->vGetState(arg2);
```

```
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetRandCF(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_dGetRandCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetRandCF" "', arg
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetRandCF" "', a
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetRandCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetRandDK(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
```

```
  if (!PyArg_ParseTuple(args,(char *)"OOO:MARKOVLV_dGetRandDK",&obj0,&obj1,&obj2)) SWIG_
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetRandDK" "', arg
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetRandDK" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetRandDK" "', a
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetRandDK(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetMeanCF(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:MARKOVLV_dGetMeanCF",&obj0,&obj1,&obj2,&obj3)
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetMeanCF" "', arg
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetMeanCF" "', a
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetMeanCF" "', a
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dGetMeanCF" "', a
  }
  arg4 = static_cast< long >(val4);
  result = (double)(arg1)->dGetMeanCF(arg2,arg3,arg4);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_dGetMeanDK(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:MARKOVLV_dGetMeanDK",&obj0,&obj1,&obj2,&obj3)
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_dGetMeanDK" "', arg
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_dGetMeanDK" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_dGetMeanDK" "', a
  }
```

```
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "MARKOVLV_dGetMeanDK" "', a
  }
  arg4 = static_cast< long >(val4);
  result = (double)(arg1)->dGetMeanDK(arg2,arg3,arg4);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vNewSeed(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_vNewSeed",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vNewSeed" "', argum
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_vNewSeed" "', arg
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vNewSeed(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vResetMeanResults(PyObject *SWIGUNUSEDPARM(self), Py
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_vResetMeanResults",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vResetMeanResults"
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  (arg1)->vResetMeanResults();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_lSeed_set(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:MARKOVLV_lSeed_set",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lSeed_set" "', argu
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "MARKOVLV_lSeed_set" "', ar
  }
  arg2 = static_cast< long >(val2);
  if (arg1) (arg1)->lSeed = arg2;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_lSeed_get(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:MARKOVLV_lSeed_get",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_lSeed_get" "', argu
```

```
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  result = (long) ((arg1)->lSeed);
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_MARKOVLV_vPrintTeX(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  MARKOVLV *arg1 = (MARKOVLV *) 0 ;
  FILE *arg2 = (FILE *) 0 ;
  bool arg3 ;
  char *arg4 = (char *) 0 ;
  bool arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  void *argp2 = 0 ;
  int res2 = 0 ;
  bool val3 ;
  int ecode3 = 0 ;
  int res4 ;
  char *buf4 = 0 ;
  int alloc4 = 0 ;
  bool val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:MARKOVLV_vPrintTeX",&obj0,&obj1,&obj2,&obj3,
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_MARKOVLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "MARKOVLV_vPrintTeX" "', argu
  }
  arg1 = reinterpret_cast< MARKOVLV * >(argp1);
  res2 = SWIG_ConvertPtr(obj1, &argp2,SWIGTYPE_p_FILE, 0 |  0 );
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "MARKOVLV_vPrintTeX" "', argu
  }
  arg2 = reinterpret_cast< FILE * >(argp2);
  ecode3 = SWIG_AsVal_bool(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "MARKOVLV_vPrintTeX" "', ar
  }
  arg3 = static_cast< bool >(val3);
  res4 = SWIG_AsCharPtrAndSize(obj3, &buf4, NULL, &alloc4);
  if (!SWIG_IsOK(res4)) {
    SWIG_exception_fail(SWIG_ArgError(res4), "in method '" "MARKOVLV_vPrintTeX" "', argu
  }
```

```
  arg4 = reinterpret_cast< char * >(buf4);
  ecode5 = SWIG_AsVal_bool(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "MARKOVLV_vPrintTeX" "', ar
  }
  arg5 = static_cast< bool >(val5);
  (arg1)->vPrintTeX(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
  if (alloc4 == SWIG_NEWOBJ) delete[] buf4;
  return resultobj;
fail:
  if (alloc4 == SWIG_NEWOBJ) delete[] buf4;
  return NULL;
}


SWIGINTERN PyObject *MARKOVLV_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_MARKOVLV, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

SWIGINTERN PyObject *_wrap_new_CAPITALLV__SWIG_0(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  CAPITALLV *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_CAPITALLV")) SWIG_fail;
  result = (CAPITALLV *)new CAPITALLV();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_CAPITALLV, SWIG_POI
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_CAPITALLV__SWIG_1(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  long arg1 ;
  CAPITALLV *result = 0 ;
  long val1 ;
  int ecode1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:new_CAPITALLV",&obj0)) SWIG_fail;
  ecode1 = SWIG_AsVal_long(obj0, &val1);
  if (!SWIG_IsOK(ecode1)) {
    SWIG_exception_fail(SWIG_ArgError(ecode1), "in method '" "new_CAPITALLV" "', argumer
  }
  arg1 = static_cast< long >(val1);
  result = (CAPITALLV *)new CAPITALLV(arg1);
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_CAPITALLV, SWIG_POI
0 );
```

```
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_CAPITALLV(PyObject *self, PyObject *args) {
  int argc;
  PyObject *argv[2];
  int ii;

  if (!PyTuple_Check(args)) SWIG_fail;
  argc = PyObject_Length(args);
  for (ii = 0; (ii < argc) && (ii < 1); ii++) {
    argv[ii] = PyTuple_GET_ITEM(args,ii);
  }
  if (argc == 0) {
    return _wrap_new_CAPITALLV__SWIG_0(self, args);
  }
  if (argc == 1) {
    int _v;
    {
      int res = SWIG_AsVal_long(argv[0], NULL);
      _v = SWIG_CheckState(res);
    }
    if (_v) {
      return _wrap_new_CAPITALLV__SWIG_1(self, args);
    }
  }

fail:
  SWIG_SetErrorMsg(PyExc_NotImplementedError,"Wrong number of arguments for overloaded f
Possible C/C++ prototypes are:\n    CAPITALLV()\n    CAPITALLV(long)\n");
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_CAPITALLV(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_CAPITALLV",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_CAPITALLV" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
```

```
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_iSetTable(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  char *arg2 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_iSetTable",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_iSetTable" "', arg
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "CAPITALLV_iSetTable" "', arg
  }
  arg2 = reinterpret_cast< char * >(buf2);
  result = (int)(arg1)->iSetTable(arg2);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetStartTime(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_vSetStartTime",&obj0,&obj1)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetStartTime" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetStartTime" '
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStartTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetStopTime(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_vSetStopTime",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetStopTime" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetStopTime" "'
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStopTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetSurvival(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
```

```
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_vSetSurvival",&obj0,&obj1,&obj2)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetSurvival" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetSurvival" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_vSetSurvival" "'
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vSetSurvival(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetDeath(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  double arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_vSetDeath",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetDeath" "', arg
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetDeath" "', a
  }
  arg2 = static_cast< double >(val2);
  (arg1)->vSetDeath(arg2);
```

```
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetPremium(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  double arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_vSetPremium",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetPremium" "', a
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetPremium" "',
  }
  arg2 = static_cast< double >(val2);
  (arg1)->vSetPremium(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetSurvivalGen(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_vSetSurvivalGen",&obj0,&obj1,&obj2))
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
```

```
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetSurvivalGen" '
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetSurvivalGen'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_vSetSurvivalGen'
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vSetSurvivalGen(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetDeathGen(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_vSetDeathGen",&obj0,&obj1,&obj2)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetDeathGen" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetDeathGen" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_vSetDeathGen" "'
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vSetDeathGen(arg2,arg3);
```

```
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_vSetPremiumGen(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_vSetPremiumGen",&obj0,&obj1,&obj2))
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_vSetPremiumGen" "'
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_vSetPremiumGen"
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_vSetPremiumGen"
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vSetPremiumGen(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetQx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetQx" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetQx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetQx" "', argu
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetFx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetFx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetFx" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
```

```
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetFx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetFx" "', argu
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetFx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_dSetBaseYear",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetBaseYear" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetBaseYear" "'
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetBaseYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double result;
```

```
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_dSetActualYear",&obj0,&obj1)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetActualYear" "'
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetActualYear"
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetDisc",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetDisc" "', argu
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetDisc" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
```

```
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetDisc" "', ar
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_dGetDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dGetDK" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dGetDK" "', argu
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_dGetCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dGetCF" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dGetCF" "', argu
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dGetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dGetQx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dGetQx" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dGetQx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dGetQx" "', argu
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetQx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
```

```
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetQx2Level(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetQx2Level",&obj0,&obj1,&obj2)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetQx2Level" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetQx2Level" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetQx2Level" "'
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQx2Level(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetSx2(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetSx2",&obj0,&obj1,&obj2)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetSx2" "', argum
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetSx2" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetSx2" "', arg
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetSx2(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetRDR(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetRDR",&obj0,&obj1,&obj2)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetRDR" "', argum
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
```

```
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetRDR" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetRDR" "', arg
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetRDR(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetSurenderPenaltyInMR(PyObject *SWIGUNUSEDPARM(se
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetSurenderPenaltyInMR",&obj0,&obj1
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetSurenderPenalt
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetSurenderPena
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetSurenderPena
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetSurenderPenaltyInMR(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
```

```
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetSHMarginInMR(PyObject *SWIGUNUSEDPARM(self), Py
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetSHMarginInMR",&obj0,&obj1,&obj2)
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetSHMarginInMR"
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetSHMarginInMR
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetSHMarginInMR
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetSHMarginInMR(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetSolaCapitalInMR(PyObject *SWIGUNUSEDPARM(self),
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
```

```
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetSolaCapitalInMR",&obj0,&obj1,&ob
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetSolaCapitalInM
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetSolaCapitall
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetSolaCapitall
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetSolaCapitalInMR(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dSetInvReturn(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:CAPITALLV_dSetInvReturn",&obj0,&obj1,&obj2)) S
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dSetInvReturn" "',
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dSetInvReturn" '
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "CAPITALLV_dSetInvReturn" '
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetInvReturn(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_CAPITALLV_dGetEV(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  CAPITALLV *arg1 = (CAPITALLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:CAPITALLV_dGetEV",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_CAPITALLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "CAPITALLV_dGetEV" "', argume
  }
  arg1 = reinterpret_cast< CAPITALLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "CAPITALLV_dGetEV" "', argu
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetEV(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *CAPITALLV_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_CAPITALLV, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}
```

```
SWIGINTERN PyObject *_wrap_new_ANNUITYLV__SWIG_0(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNUITYLV *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_ANNUITYLV")) SWIG_fail;
  result = (ANNUITYLV *)new ANNUITYLV();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_ANNUITYLV, SWIG_POI
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_ANNUITYLV__SWIG_1(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  long arg1 ;
  double arg2 ;
  ANNUITYLV *result = 0 ;
  long val1 ;
  int ecode1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:new_ANNUITYLV",&obj0,&obj1)) SWIG_fail;
  ecode1 = SWIG_AsVal_long(obj0, &val1);
  if (!SWIG_IsOK(ecode1)) {
    SWIG_exception_fail(SWIG_ArgError(ecode1), "in method '" "new_ANNUITYLV" "', argumer
  }
  arg1 = static_cast< long >(val1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "new_ANNUITYLV" "', argumer
  }
  arg2 = static_cast< double >(val2);
  result = (ANNUITYLV *)new ANNUITYLV(arg1,arg2);
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_ANNUITYLV, SWIG_POI
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_ANNUITYLV(PyObject *self, PyObject *args) {
  int argc;
  PyObject *argv[3];
  int ii;

  if (!PyTuple_Check(args)) SWIG_fail;
  argc = PyObject_Length(args);
  for (ii = 0; (ii < argc) && (ii < 2); ii++) {
```

```
    argv[ii] = PyTuple_GET_ITEM(args,ii);
  }
  if (argc == 0) {
    return _wrap_new_ANNUITYLV__SWIG_0(self, args);
  }
  if (argc == 2) {
    int _v;
    {
      int res = SWIG_AsVal_long(argv[0], NULL);
      _v = SWIG_CheckState(res);
    }
    if (_v) {
      {
        int res = SWIG_AsVal_double(argv[1], NULL);
        _v = SWIG_CheckState(res);
      }
      if (_v) {
        return _wrap_new_ANNUITYLV__SWIG_1(self, args);
      }
    }
  }

fail:
  SWIG_SetErrorMsg(PyExc_NotImplementedError,"Wrong number of arguments for overloaded f
Possible C/C++ prototypes are:\n    ANNUITYLV()\n    ANNUITYLV(long,double)\n");
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_ANNUITYLV(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_ANNUITYLV",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_ANNUITYLV" "', argume
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_iSetTable(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
```

```
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  char *arg2 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_iSetTable",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_iSetTable" "', arg
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "ANNUITYLV_iSetTable" "', arg
  }
  arg2 = reinterpret_cast< char * >(buf2);
  result = (int)(arg1)->iSetTable(arg2);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetStartTime(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_vSetStartTime",&obj0,&obj1)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetStartTime" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetStartTime" '
  }
  arg2 = static_cast< long >(val2);
```

```
  (arg1)->vSetStartTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetStopTime(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_vSetStopTime",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetStopTime" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetStopTime" "'
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStopTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetSAge(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_vSetSAge",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetSAge" "', argu
  }
```

```
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetSAge" "', ar
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetSAge(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetG(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_vSetG",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetG" "', argumer
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetG" "', argum
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetG(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetMaxProj(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_vSetMaxProj",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetMaxProj" "', a
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetMaxProj" "',
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetMaxProj(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV_dSetQx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetQx" "', argume
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetQx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_dSetQx" "', argu
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
```

```
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetFx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV_dSetFx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetFx" "', argume
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetFx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_dSetFx" "', argu
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetFx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetSx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV_dSetSx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetSx" "', argume
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetSx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_dSetSx" "', argu
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetSx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dSetBaseYear",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetBaseYear" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetBaseYear" "'
  }
```

```
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetBaseYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dSetActualYear",&obj0,&obj1)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetActualYear" "'
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetActualYear"
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV_dSetDisc",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetDisc" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetDisc" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_dSetDisc" "', ar
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dGetDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dGetDK" "', argume
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dGetDK" "', argu
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dGetCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dGetCF" "', argume
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dGetCF" "', argu
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dGetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV_dGetQx",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dGetQx" "', argume
```

```
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dGetQx" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_dGetQx" "', argu
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetQx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dGetTqx(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dGetTqx",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dGetTqx" "', argum
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dGetTqx" "', arg
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetTqx(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dGetTpx(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
```

```
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dGetTpx",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dGetTpx" "', argum
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dGetTpx" "', arg
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetTpx(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetPre(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  double arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV_dSetPre",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetPre" "', argum
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetPre" "', arg
  }
  arg2 = static_cast< double >(val2);
  result = (double)(arg1)->dSetPre(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
```

```
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_dSetRelativeQxForTime(PyObject *SWIGUNUSEDPARM(self
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV_dSetRelativeQxForTime",&obj0,&obj1,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_dSetRelativeQxForT
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_dSetRelativeQxFo
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_dSetRelativeQxFo
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetRelativeQxForTime(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vLeistReset(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:ANNUITYLV_vLeistReset",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
```

```
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vLeistReset" "', a
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  (arg1)->vLeistReset();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetLeistLinear(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double arg4 ;
  double arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:ANNUITYLV_vSetLeistLinear",&obj0,&obj1,&obj2
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetLeistLinear" '
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetLeistLinear'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_vSetLeistLinear'
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNUITYLV_vSetLeistLinear'
```

```
  }
  arg4 = static_cast< double >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNUITYLV_vSetLeistLinear'
  }
  arg5 = static_cast< double >(val5);
  (arg1)->vSetLeistLinear(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV_vSetLeistExp(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNUITYLV *arg1 = (ANNUITYLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double arg4 ;
  double arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:ANNUITYLV_vSetLeistExp",&obj0,&obj1,&obj2,&o
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV_vSetLeistExp" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV_vSetLeistExp" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV_vSetLeistExp" "'
  }
  arg3 = static_cast< long >(val3);
```

```
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNUITYLV_vSetLeistExp" "'
  }
  arg4 = static_cast< double >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNUITYLV_vSetLeistExp" "'
  }
  arg5 = static_cast< double >(val5);
  (arg1)->vSetLeistExp(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *ANNUITYLV_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_ANNUITYLV, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

SWIGINTERN PyObject *_wrap_new_ANNUITYLV2__SWIG_0(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNUITYLV2 *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_ANNUITYLV2")) SWIG_fail;
  result = (ANNUITYLV2 *)new ANNUITYLV2();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_ANNUITYLV2, SWIG_PO
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_ANNUITYLV2__SWIG_1(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  long arg1 ;
  double arg2 ;
  ANNUITYLV2 *result = 0 ;
  long val1 ;
  int ecode1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:new_ANNUITYLV2",&obj0,&obj1)) SWIG_fail;
  ecode1 = SWIG_AsVal_long(obj0, &val1);
  if (!SWIG_IsOK(ecode1)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode1), "in method '" "new_ANNUITYLV2" "', argume
  }
  arg1 = static_cast< long >(val1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "new_ANNUITYLV2" "', argume
  }
  arg2 = static_cast< double >(val2);
  result = (ANNUITYLV2 *)new ANNUITYLV2(arg1,arg2);
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_ANNUITYLV2, SWIG_PO
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_ANNUITYLV2(PyObject *self, PyObject *args) {
  int argc;
  PyObject *argv[3];
  int ii;

  if (!PyTuple_Check(args)) SWIG_fail;
  argc = PyObject_Length(args);
  for (ii = 0; (ii < argc) && (ii < 2); ii++) {
    argv[ii] = PyTuple_GET_ITEM(args,ii);
  }
  if (argc == 0) {
    return _wrap_new_ANNUITYLV2__SWIG_0(self, args);
  }
  if (argc == 2) {
    int _v;
    {
      int res = SWIG_AsVal_long(argv[0], NULL);
      _v = SWIG_CheckState(res);
    }
    if (_v) {
      {
        int res = SWIG_AsVal_double(argv[1], NULL);
        _v = SWIG_CheckState(res);
      }
      if (_v) {
        return _wrap_new_ANNUITYLV2__SWIG_1(self, args);
      }
    }
  }

fail:
  SWIG_SetErrorMsg(PyExc_NotImplementedError,"Wrong number of arguments for overloaded f
Possible C/C++ prototypes are:\n    ANNUITYLV2()\n    ANNUITYLV2(long,double)\n");
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_delete_ANNUITYLV2(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_ANNUITYLV2",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_ANNUITYLV2" "', argum
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_iSetTable1(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  char *arg2 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_iSetTable1",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_iSetTable1" "', a
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "ANNUITYLV2_iSetTable1" "', a
  }
  arg2 = reinterpret_cast< char * >(buf2);
  result = (int)(arg1)->iSetTable1(arg2);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_iSetTable2(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  char *arg2 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_iSetTable2",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_iSetTable2" "', a
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "ANNUITYLV2_iSetTable2" "', a
  }
  arg2 = reinterpret_cast< char * >(buf2);
  result = (int)(arg1)->iSetTable2(arg2);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_vSetStartTime(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_vSetStartTime",&obj0,&obj1)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vSetStartTime" "'
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
```

```
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_vSetStartTime"
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStartTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_vSetStopTime(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_vSetStopTime",&obj0,&obj1)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vSetStopTime" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_vSetStopTime" '
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStopTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_vSetSAge1(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_vSetSAge1",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vSetSAge1" "', ar
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_vSetSAge1" "',
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetSAge1(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_vSetSAge2(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_vSetSAge2",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vSetSAge2" "', ar
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_vSetSAge2" "',
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetSAge2(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetQx1(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double arg3 ;
```

```
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetQx1",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetQx1" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetQx1" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetQx1" "', ar
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQx1(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetFx1(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetFx1",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetFx1" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetFx1" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetFx1" "', ar
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetFx1(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetQx2(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetQx2",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetQx2" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetQx2" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetQx2" "', ar
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQx2(arg2,arg3);
```

```
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetFx2(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetFx2",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetFx2" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetFx2" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetFx2" "', ar
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetFx2(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_dSetBaseYear",&obj0,&obj1)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetBaseYear" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetBaseYear" '
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetBaseYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_dSetActualYear",&obj0,&obj1)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetActualYear" '
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetActualYear'
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetDisc",&obj0,&obj1,&obj2)) SWIG_
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetDisc" "', arg
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetDisc" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetDisc" "', a
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
```

```
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dGetDK",&obj0,&obj1,&obj2)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dGetDK" "', argum
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dGetDK" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dGetDK" "', arg
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetDK(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_dGetCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dGetCF" "', argum
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dGetCF" "', arg
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
```

```
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dGetQx1(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dGetQx1",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dGetQx1" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dGetQx1" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dGetQx1" "', ar
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetQx1(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dGetQx2(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
```

```
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dGetQx2",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dGetQx2" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dGetQx2" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dGetQx2" "', ar
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetQx2(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetY_Minus_X(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetY_Minus_X",&obj0,&obj1,&obj2))
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetY_Minus_X" "'
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetY_Minus_X"
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetY_Minus_X"
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dSetY_Minus_X(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetBenefit(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNUITYLV2_dSetBenefit",&obj0,&obj1,&obj2)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetBenefit" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetBenefit" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_dSetBenefit" "'
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetBenefit(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_ANNUITYLV2_dSetPre(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  double arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNUITYLV2_dSetPre",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_dSetPre" "', argu
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_dSetPre" "', ar
  }
  arg2 = static_cast< double >(val2);
  result = (double)(arg1)->dSetPre(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_vLeistReset(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:ANNUITYLV2_vLeistReset",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vLeistReset" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  (arg1)->vLeistReset();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_ANNUITYLV2_vSetLeistLinear(PyObject *SWIGUNUSEDPARM(self), Py
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  long arg3 ;
  double arg4 ;
  double arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:ANNUITYLV2_vSetLeistLinear",&obj0,&obj1,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vSetLeistLinear"
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_vSetLeistLinear
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_vSetLeistLinear
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNUITYLV2_vSetLeistLinear
  }
  arg4 = static_cast< double >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNUITYLV2_vSetLeistLinear
  }
  arg5 = static_cast< double >(val5);
  (arg1)->vSetLeistLinear(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_ANNUITYLV2_vSetLeistExp(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  ANNUITYLV2 *arg1 = (ANNUITYLV2 *) 0 ;
  long arg2 ;
  long arg3 ;
  double arg4 ;
  double arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:ANNUITYLV2_vSetLeistExp",&obj0,&obj1,&obj2,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNUITYLV2, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNUITYLV2_vSetLeistExp" "',
  }
  arg1 = reinterpret_cast< ANNUITYLV2 * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNUITYLV2_vSetLeistExp" '
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNUITYLV2_vSetLeistExp" '
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNUITYLV2_vSetLeistExp" '
  }
  arg4 = static_cast< double >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNUITYLV2_vSetLeistExp" '
  }
  arg5 = static_cast< double >(val5);
  (arg1)->vSetLeistExp(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
```

```
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *ANNUITYLV2_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_ANNUITYLV2, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

SWIGINTERN PyObject *_wrap_new_WIDDOWLV__SWIG_0(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  WIDDOWLV *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_WIDDOWLV")) SWIG_fail;
  result = (WIDDOWLV *)new WIDDOWLV();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_WIDDOWLV, SWIG_POIN
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_WIDDOWLV__SWIG_1(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  long arg1 ;
  WIDDOWLV *result = 0 ;
  long val1 ;
  int ecode1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:new_WIDDOWLV",&obj0)) SWIG_fail;
  ecode1 = SWIG_AsVal_long(obj0, &val1);
  if (!SWIG_IsOK(ecode1)) {
    SWIG_exception_fail(SWIG_ArgError(ecode1), "in method '" "new_WIDDOWLV" "', argument
  }
  arg1 = static_cast< long >(val1);
  result = (WIDDOWLV *)new WIDDOWLV(arg1);
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_WIDDOWLV, SWIG_POIN
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_new_WIDDOWLV(PyObject *self, PyObject *args) {
  int argc;
  PyObject *argv[2];
  int ii;
```

```
  if (!PyTuple_Check(args)) SWIG_fail;
  argc = PyObject_Length(args);
  for (ii = 0; (ii < argc) && (ii < 1); ii++) {
    argv[ii] = PyTuple_GET_ITEM(args,ii);
  }
  if (argc == 0) {
    return _wrap_new_WIDDOWLV__SWIG_0(self, args);
  }
  if (argc == 1) {
    int _v;
    {
      int res = SWIG_AsVal_long(argv[0], NULL);
      _v = SWIG_CheckState(res);
    }
    if (_v) {
      return _wrap_new_WIDDOWLV__SWIG_1(self, args);
    }
  }

fail:
  SWIG_SetErrorMsg(PyExc_NotImplementedError,"Wrong number of arguments for overloaded f
Possible C/C++ prototypes are:\n    WIDDOWLV()\n    WIDDOWLV(long)\n");
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_WIDDOWLV(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_WIDDOWLV",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_WIDDOWLV" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_vSetStartTime(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
```

```
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_vSetStartTime",&obj0,&obj1)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_vSetStartTime" "',
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_vSetStartTime" "'
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStartTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_vSetStopTime(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_vSetStopTime",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_vSetStopTime" "', a
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_vSetStopTime" "',
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vSetStopTime(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetQx",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetQx" "', argumen
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetQx" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetQx" "', argum
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetQy(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
```

```
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetQy",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetQy" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetQy" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetQy" "', argum
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetQy(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetFx(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetFx",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetFx" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetFx" "', argum
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetFx" "', argum
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetFx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetFy(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetFy",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetFy" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetFy" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetFy" "', argum
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetFy(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetHx(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetHx",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetHx" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetHx" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetHx" "', argum
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetHx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetYx(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetYx",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetYx" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetYx" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetYx" "', argum
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetYx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_dSetBaseYear",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetBaseYear" "', a
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetBaseYear" "',
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetBaseYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_dSetActualYear",&obj0,&obj1)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetActualYear" "',
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetActualYear" '
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dSetDisc",&obj0,&obj1,&obj2)) SWIG_fa
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetDisc" "', argum
```

```
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetDisc" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dSetDisc" "', arg
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_dGetDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dGetDK" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dGetDK" "', argum
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
```

```
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_dGetCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dGetCF" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dGetCF" "', argum
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dGetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:WIDDOWLV_dGetQx",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dGetQx" "', argumer
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dGetQx" "', argum
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_dGetQx" "', argum
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetQx(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_dSetPre(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  double arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  double val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:WIDDOWLV_dSetPre",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_dSetPre" "', argume
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_double(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_dSetPre" "', argu
  }
  arg2 = static_cast< double >(val2);
  result = (double)(arg1)->dSetPre(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_vLeistReset(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:WIDDOWLV_vLeistReset",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
```

```
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_vLeistReset" "', ar
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  (arg1)->vLeistReset();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_vSetLeistLinear(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double arg4 ;
  double arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:WIDDOWLV_vSetLeistLinear",&obj0,&obj1,&obj2,
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_vSetLeistLinear" "'
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_vSetLeistLinear"
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_vSetLeistLinear"
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "WIDDOWLV_vSetLeistLinear"
```

```
  }
  arg4 = static_cast< double >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "WIDDOWLV_vSetLeistLinear"
  }
  arg5 = static_cast< double >(val5);
  (arg1)->vSetLeistLinear(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_WIDDOWLV_vSetLeistExp(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  WIDDOWLV *arg1 = (WIDDOWLV *) 0 ;
  long arg2 ;
  long arg3 ;
  double arg4 ;
  double arg5 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:WIDDOWLV_vSetLeistExp",&obj0,&obj1,&obj2,&ob
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_WIDDOWLV, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "WIDDOWLV_vSetLeistExp" "', a
  }
  arg1 = reinterpret_cast< WIDDOWLV * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "WIDDOWLV_vSetLeistExp" "',
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "WIDDOWLV_vSetLeistExp" "',
  }
  arg3 = static_cast< long >(val3);
```

```
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "WIDDOWLV_vSetLeistExp" "',
  }
  arg4 = static_cast< double >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "WIDDOWLV_vSetLeistExp" "',
  }
  arg5 = static_cast< double >(val5);
  (arg1)->vSetLeistExp(arg2,arg3,arg4,arg5);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *WIDDOWLV_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_WIDDOWLV, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

SWIGINTERN PyObject *_wrap_new_GLMOD(PyObject *SWIGUNUSEDPARM(self), PyObject *args) {
  PyObject *resultobj = 0;
  GLMOD *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_GLMOD")) SWIG_fail;
  result = (GLMOD *)new GLMOD();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_GLMOD, SWIG_POINTER
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_GLMOD(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_GLMOD",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_GLMOD" "', argument '
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  delete arg1;
```

```
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:GLMOD_dSetQx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetQx" "', argument '
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetQx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dSetQx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_dSetQx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_dSetQx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_dSetQx" "', argument
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetQx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetFx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:GLMOD_dSetFx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetFx" "', argument '
  }
```

```
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetFx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dSetFx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_dSetFx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_dSetFx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_dSetFx" "', argument
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetFx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetSx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
```

```
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:GLMOD_dSetSx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetSx" "', argument "
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetSx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dSetSx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_dSetSx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_dSetSx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_dSetSx" "', argument
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetSx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
```

```
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:GLMOD_dSetBaseYear",&obj0,&obj1,&obj2,&obj3,
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetBaseYear" "', argu
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetBaseYear" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dSetBaseYear" "', ar
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_dSetBaseYear" "', ar
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_dSetBaseYear" "', ar
  }
  arg5 = static_cast< long >(val5);
  result = (double)(arg1)->dSetBaseYear(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
```

```
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:GLMOD_dSetActualYear",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetActualYear" "', ar
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetActualYear" "',
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *args
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_dSetDisc",&obj0,&obj1,&obj2)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetDisc" "', argument
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetDisc" "', argume
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dSetDisc" "', argume
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vStress(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_vStress",&obj0,&obj1,&obj2)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vStress" "', argument
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_vStress" "', argumer
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_vStress" "', argumer
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vStress(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vAddAnnuity(PyObject *SWIGUNUSEDPARM(self), PyObject *a
```

```
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOO:GLMOD_vAddAnnuity",&obj0,&obj1,&obj2,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vAddAnnuity" "', argum
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
```

```
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "GLMOD_vAddAnnuity" "', arg
  }
  arg8 = static_cast< double >(val8);
  (arg1)->vAddAnnuity(arg2,arg3,arg4,arg5,arg6,arg7,arg8);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vAddEndowment(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  double arg5 ;
  double arg6 ;
  double arg7 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOO:GLMOD_vAddEndowment",&obj0,&obj1,&obj2,&ob
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vAddEndowment" "', arg
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_vAddEndowment" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_vAddEndowment" "', a
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_vAddEndowment" "', a
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_vAddEndowment" "', a
  }
  arg5 = static_cast< double >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_vAddEndowment" "', a
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "GLMOD_vAddEndowment" "', a
  }
  arg7 = static_cast< double >(val7);
  (arg1)->vAddEndowment(arg2,arg3,arg4,arg5,arg6,arg7);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vAddWiddow(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
```

```
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOO:GLMOD_vAddWiddow",&obj0,&obj1,&obj2,&obj3
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vAddWiddow" "', argume
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "GLMOD_vAddWiddow" "', argu
  }
  arg8 = static_cast< double >(val8);
  (arg1)->vAddWiddow(arg2,arg3,arg4,arg5,arg6,arg7,arg8);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vSetRKWAnnuity(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_vSetRKWAnnuity",&obj0,&obj1,&obj2)) SWIG
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vSetRKWAnnuity" "', ar
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_vSetRKWAnnuity" "',
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_vSetRKWAnnuity" "',
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vSetRKWAnnuity(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vSetRKWEndowment(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_vSetRKWEndowment",&obj0,&obj1,&obj2)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vSetRKWEndowment" "',
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_vSetRKWEndowment" "'
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_vSetRKWEndowment" "'
  }
  arg3 = static_cast< double >(val3);
  (arg1)->vSetRKWEndowment(arg2,arg3);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vUpdateOperator(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  void *argp1 = 0 ;
```

```
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:GLMOD_vUpdateOperator",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vUpdateOperator" "', a
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  (arg1)->vUpdateOperator();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:GLMOD_dGetDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetDK" "', argument '
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetDK" "', argument
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetDKDetail(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
```

```
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_dGetDKDetail",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetDKDetail" "', argu
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetDKDetail" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dGetDKDetail" "', ar
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetDKDetail(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetDKTilde(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:GLMOD_dGetDKTilde",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetDKTilde" "', argum
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetDKTilde" "', arg
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetDKTilde(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetStatDK__SWIG_0(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:GLMOD_dGetStatDK",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetStatDK" "', argume
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  result = (double)(arg1)->dGetStatDK();
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetFVDK(PyObject *SWIGUNUSEDPARM(self), PyObject *args
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:GLMOD_dGetFVDK",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetFVDK" "', argument
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  result = (double)(arg1)->dGetFVDK();
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_GLMOD_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:GLMOD_dGetCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetCF" "', argument '
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetCF" "', argument
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetCFDetail(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_dGetCFDetail",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetCFDetail" "', argu
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
```

```
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetCFDetail" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dGetCFDetail" "', ar
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetCFDetail(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetStatDK__SWIG_1(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:GLMOD_dGetStatDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetStatDK" "', argume
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetStatDK" "', argu
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetStatDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetStatDK(PyObject *self, PyObject *args) {
  int argc;
  PyObject *argv[3];
  int ii;
```

```
  if (!PyTuple_Check(args)) SWIG_fail;
  argc = PyObject_Length(args);
  for (ii = 0; (ii < argc) && (ii < 2); ii++) {
    argv[ii] = PyTuple_GET_ITEM(args,ii);
  }
  if (argc == 1) {
    int _v;
    void *vptr = 0;
    int res = SWIG_ConvertPtr(argv[0], &vptr, SWIGTYPE_p_GLMOD, 0);
    _v = SWIG_CheckState(res);
    if (_v) {
      return _wrap_GLMOD_dGetStatDK__SWIG_0(self, args);
    }
  }
  if (argc == 2) {
    int _v;
    void *vptr = 0;
    int res = SWIG_ConvertPtr(argv[0], &vptr, SWIGTYPE_p_GLMOD, 0);
    _v = SWIG_CheckState(res);
    if (_v) {
      {
        int res = SWIG_AsVal_long(argv[1], NULL);
        _v = SWIG_CheckState(res);
      }
      if (_v) {
        return _wrap_GLMOD_dGetStatDK__SWIG_1(self, args);
      }
    }
  }

fail:
  SWIG_SetErrorMsg(PyExc_NotImplementedError,"Wrong number of arguments for overloaded f
Possible C/C++ prototypes are:\n    dGetStatDK()\n    dGetStatDK(long)\n");
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dGetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
```

```
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:GLMOD_dGetQx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dGetQx" "', argument "
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dGetQx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dGetQx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "GLMOD_dGetQx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "GLMOD_dGetQx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "GLMOD_dGetQx" "', argument
  }
  arg6 = static_cast< long >(val6);
  result = (double)(arg1)->dGetQx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_dSetRelativeQxForTime(PyObject *SWIGUNUSEDPARM(self), P
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  long arg2 ;
```

```
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:GLMOD_dSetRelativeQxForTime",&obj0,&obj1,&obj2
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_dSetRelativeQxForTime'
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_dSetRelativeQxForTim
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_dSetRelativeQxForTim
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetRelativeQxForTime(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_iReadInforce(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  int arg2 ;
  int arg3 ;
  char *arg4 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int val2 ;
  int ecode2 = 0 ;
  int val3 ;
  int ecode3 = 0 ;
  int res4 ;
  char *buf4 = 0 ;
  int alloc4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:GLMOD_iReadInforce",&obj0,&obj1,&obj2,&obj3))
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_iReadInforce" "', argu
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  ecode2 = SWIG_AsVal_int(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "GLMOD_iReadInforce" "', ar
  }
  arg2 = static_cast< int >(val2);
  ecode3 = SWIG_AsVal_int(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "GLMOD_iReadInforce" "', ar
  }
  arg3 = static_cast< int >(val3);
  res4 = SWIG_AsCharPtrAndSize(obj3, &buf4, NULL, &alloc4);
  if (!SWIG_IsOK(res4)) {
    SWIG_exception_fail(SWIG_ArgError(res4), "in method '" "GLMOD_iReadInforce" "', argu
  }
  arg4 = reinterpret_cast< char * >(buf4);
  result = (int)(arg1)->iReadInforce(arg2,arg3,arg4);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc4 == SWIG_NEWOBJ) delete[] buf4;
  return resultobj;
fail:
  if (alloc4 == SWIG_NEWOBJ) delete[] buf4;
  return NULL;
}


SWIGINTERN PyObject *_wrap_GLMOD_vPrintTex(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  GLMOD *arg1 = (GLMOD *) 0 ;
  char *arg2 = (char *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:GLMOD_vPrintTex",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_GLMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "GLMOD_vPrintTex" "', argumer
  }
  arg1 = reinterpret_cast< GLMOD * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "GLMOD_vPrintTex" "', argumer
  }
  arg2 = reinterpret_cast< char * >(buf2);
  (arg1)->vPrintTex(arg2);
  resultobj = SWIG_Py_Void();
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *GLMOD_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_GLMOD, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

SWIGINTERN PyObject *_wrap_new_ANNMOD(PyObject *SWIGUNUSEDPARM(self), PyObject *args) {
  PyObject *resultobj = 0;
  ANNMOD *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_ANNMOD")) SWIG_fail;
  result = (ANNMOD *)new ANNMOD();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_ANNMOD, SWIG_POINTE
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_ANNMOD(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_ANNMOD",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_ANNMOD" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:ANNMOD_dSetQx",&obj0,&obj1,&obj2,&obj3,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetQx" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetQx" "', argumer
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dSetQx" "', argumer
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_dSetQx" "', argumer
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_dSetQx" "', argumer
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_dSetQx" "', argumer
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetQx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetFx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:ANNMOD_dSetFx",&obj0,&obj1,&obj2,&obj3,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetFx" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetFx" "', argumer
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dSetFx" "', argumer
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_dSetFx" "', argumer
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_dSetFx" "', argumer
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_dSetFx" "', argumer
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetFx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetSx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
```

```
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:ANNMOD_dSetSx",&obj0,&obj1,&obj2,&obj3,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetSx" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetSx" "', argumen
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dSetSx" "', argumen
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_dSetSx" "', argumen
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_dSetSx" "', argumen
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_dSetSx" "', argumen
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetSx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
```

```
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:ANNMOD_dSetBaseYear",&obj0,&obj1,&obj2,&obj3
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetBaseYear" "', arg
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetBaseYear" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dSetBaseYear" "', a
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_dSetBaseYear" "', a
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_dSetBaseYear" "', a
  }
  arg5 = static_cast< long >(val5);
  result = (double)(arg1)->dSetBaseYear(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
```

```
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNMOD_dSetActualYear",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetActualYear" "', a
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetActualYear" "',
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNMOD_dSetDisc",&obj0,&obj1,&obj2)) SWIG_fail
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetDisc" "', argumer
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetDisc" "', argum
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dSetDisc" "', argum
  }
```

```
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vAddAnnuity1(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  double arg9 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  double val9 ;
  int ecode9 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;
  PyObject * obj8 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOOO:ANNMOD_vAddAnnuity1",&obj0,&obj1,&obj2,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vAddAnnuity1" "', arg
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
```

```
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg8 = static_cast< double >(val8);
  ecode9 = SWIG_AsVal_double(obj8, &val9);
  if (!SWIG_IsOK(ecode9)) {
    SWIG_exception_fail(SWIG_ArgError(ecode9), "in method '" "ANNMOD_vAddAnnuity1" "', a
  }
  arg9 = static_cast< double >(val9);
  (arg1)->vAddAnnuity1(arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vAddAnnuity0(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
```

```
  double arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  double val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOO:ANNMOD_vAddAnnuity0",&obj0,&obj1,&obj2,&o
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vAddAnnuity0" "', arg
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_double(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
```

```
  arg5 = static_cast< double >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "ANNMOD_vAddAnnuity0" "', a
  }
  arg8 = static_cast< double >(val8);
  (arg1)->vAddAnnuity0(arg2,arg3,arg4,arg5,arg6,arg7,arg8);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vAddAnnuity2xy(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  long arg7 ;
  double arg8 ;
  double arg9 ;
  double arg10 ;
  double arg11 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
  int ecode6 = 0 ;
  long val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  double val9 ;
```

```
  int ecode9 = 0 ;
  double val10 ;
  int ecode10 = 0 ;
  double val11 ;
  int ecode11 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;
  PyObject * obj8 = 0 ;
  PyObject * obj9 = 0 ;
  PyObject * obj10 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOOOOO:ANNMOD_vAddAnnuity2xy",&obj0,&obj1,&ob
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vAddAnnuity2xy" "', a
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg6 = static_cast< long >(val6);
  ecode7 = SWIG_AsVal_long(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg7 = static_cast< long >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
```

```
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg8 = static_cast< double >(val8);
  ecode9 = SWIG_AsVal_double(obj8, &val9);
  if (!SWIG_IsOK(ecode9)) {
    SWIG_exception_fail(SWIG_ArgError(ecode9), "in method '" "ANNMOD_vAddAnnuity2xy" "',
  }
  arg9 = static_cast< double >(val9);
  ecode10 = SWIG_AsVal_double(obj9, &val10);
  if (!SWIG_IsOK(ecode10)) {
    SWIG_exception_fail(SWIG_ArgError(ecode10), "in method '" "ANNMOD_vAddAnnuity2xy" "'
  }
  arg10 = static_cast< double >(val10);
  ecode11 = SWIG_AsVal_double(obj10, &val11);
  if (!SWIG_IsOK(ecode11)) {
    SWIG_exception_fail(SWIG_ArgError(ecode11), "in method '" "ANNMOD_vAddAnnuity2xy" "'
  }
  arg11 = static_cast< double >(val11);
  (arg1)->vAddAnnuity2xy(arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10,arg11);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vAddAnnuity2xyBar(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  long arg7 ;
  double arg8 ;
  double arg9 ;
  double arg10 ;
  double arg11 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
  int ecode6 = 0 ;
  long val7 ;
  int ecode7 = 0 ;
```

```
  double val8 ;
  int ecode8 = 0 ;
  double val9 ;
  int ecode9 = 0 ;
  double val10 ;
  int ecode10 = 0 ;
  double val11 ;
  int ecode11 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;
  PyObject * obj8 = 0 ;
  PyObject * obj9 = 0 ;
  PyObject * obj10 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOOOOO:ANNMOD_vAddAnnuity2xyBar",&obj0,&obj1,
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vAddAnnuity2xyBar" "'
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg6 = static_cast< long >(val6);
  ecode7 = SWIG_AsVal_long(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "ANNMOD_vAddAnnuity2xyBar"
```

```
  }
  arg7 = static_cast< long >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg8 = static_cast< double >(val8);
  ecode9 = SWIG_AsVal_double(obj8, &val9);
  if (!SWIG_IsOK(ecode9)) {
    SWIG_exception_fail(SWIG_ArgError(ecode9), "in method '" "ANNMOD_vAddAnnuity2xyBar"
  }
  arg9 = static_cast< double >(val9);
  ecode10 = SWIG_AsVal_double(obj9, &val10);
  if (!SWIG_IsOK(ecode10)) {
    SWIG_exception_fail(SWIG_ArgError(ecode10), "in method '" "ANNMOD_vAddAnnuity2xyBar'
  }
  arg10 = static_cast< double >(val10);
  ecode11 = SWIG_AsVal_double(obj10, &val11);
  if (!SWIG_IsOK(ecode11)) {
    SWIG_exception_fail(SWIG_ArgError(ecode11), "in method '" "ANNMOD_vAddAnnuity2xyBar'
  }
  arg11 = static_cast< double >(val11);
  (arg1)->vAddAnnuity2xyBar(arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10,arg11);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vAddAnnuity2xToy(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  long arg7 ;
  double arg8 ;
  double arg9 ;
  double arg10 ;
  double arg11 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
```

```
  int ecode6 = 0 ;
  long val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  double val9 ;
  int ecode9 = 0 ;
  double val10 ;
  int ecode10 = 0 ;
  double val11 ;
  int ecode11 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;
  PyObject * obj8 = 0 ;
  PyObject * obj9 = 0 ;
  PyObject * obj10 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOOOOO:ANNMOD_vAddAnnuity2xToy",&obj0,&obj1,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vAddAnnuity2xToy" "',
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg6 = static_cast< long >(val6);
```

```
  ecode7 = SWIG_AsVal_long(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg7 = static_cast< long >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg8 = static_cast< double >(val8);
  ecode9 = SWIG_AsVal_double(obj8, &val9);
  if (!SWIG_IsOK(ecode9)) {
    SWIG_exception_fail(SWIG_ArgError(ecode9), "in method '" "ANNMOD_vAddAnnuity2xToy" '
  }
  arg9 = static_cast< double >(val9);
  ecode10 = SWIG_AsVal_double(obj9, &val10);
  if (!SWIG_IsOK(ecode10)) {
    SWIG_exception_fail(SWIG_ArgError(ecode10), "in method '" "ANNMOD_vAddAnnuity2xToy"
  }
  arg10 = static_cast< double >(val10);
  ecode11 = SWIG_AsVal_double(obj10, &val11);
  if (!SWIG_IsOK(ecode11)) {
    SWIG_exception_fail(SWIG_ArgError(ecode11), "in method '" "ANNMOD_vAddAnnuity2xToy"
  }
  arg11 = static_cast< double >(val11);
  (arg1)->vAddAnnuity2xToy(arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10,arg11);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vAddAnnuity2yTox(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  long arg7 ;
  double arg8 ;
  double arg9 ;
  double arg10 ;
  double arg11 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
```

```
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
  int ecode6 = 0 ;
  long val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  double val9 ;
  int ecode9 = 0 ;
  double val10 ;
  int ecode10 = 0 ;
  double val11 ;
  int ecode11 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;
  PyObject * obj8 = 0 ;
  PyObject * obj9 = 0 ;
  PyObject * obj10 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOOOOO:ANNMOD_vAddAnnuity2yTox",&obj0,&obj1,&
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vAddAnnuity2yTox" "',
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg6 = static_cast< long >(val6);
  ecode7 = SWIG_AsVal_long(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg7 = static_cast< long >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg8 = static_cast< double >(val8);
  ecode9 = SWIG_AsVal_double(obj8, &val9);
  if (!SWIG_IsOK(ecode9)) {
    SWIG_exception_fail(SWIG_ArgError(ecode9), "in method '" "ANNMOD_vAddAnnuity2yTox" '
  }
  arg9 = static_cast< double >(val9);
  ecode10 = SWIG_AsVal_double(obj9, &val10);
  if (!SWIG_IsOK(ecode10)) {
    SWIG_exception_fail(SWIG_ArgError(ecode10), "in method '" "ANNMOD_vAddAnnuity2yTox"
  }
  arg10 = static_cast< double >(val10);
  ecode11 = SWIG_AsVal_double(obj10, &val11);
  if (!SWIG_IsOK(ecode11)) {
    SWIG_exception_fail(SWIG_ArgError(ecode11), "in method '" "ANNMOD_vAddAnnuity2yTox"
  }
  arg11 = static_cast< double >(val11);
  (arg1)->vAddAnnuity2yTox(arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10,arg11);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_vUpdateOperator(PyObject *SWIGUNUSEDPARM(self), PyObje
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:ANNMOD_vUpdateOperator",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_vUpdateOperator" "',
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  (arg1)->vUpdateOperator();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
```

```
}


SWIGINTERN PyObject *_wrap_ANNMOD_dGetDK(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNMOD_dGetDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dGetDK" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dGetDK" "', argumer
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dGetStatDK__SWIG_0(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:ANNMOD_dGetStatDK",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dGetStatDK" "', argum
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  result = (double)(arg1)->dGetStatDK();
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_ANNMOD_dGetFVDK(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:ANNMOD_dGetFVDK",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dGetFVDK" "', argumer
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  result = (double)(arg1)->dGetFVDK();
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dGetCF(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNMOD_dGetCF",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dGetCF" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dGetCF" "', argumer
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetCF(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_ANNMOD_dGetStatDK__SWIG_1(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:ANNMOD_dGetStatDK",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dGetStatDK" "', argum
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dGetStatDK" "', arg
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dGetStatDK(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dGetStatDK(PyObject *self, PyObject *args) {
  int argc;
  PyObject *argv[3];
  int ii;

  if (!PyTuple_Check(args)) SWIG_fail;
  argc = PyObject_Length(args);
  for (ii = 0; (ii < argc) && (ii < 2); ii++) {
    argv[ii] = PyTuple_GET_ITEM(args,ii);
  }
  if (argc == 1) {
    int _v;
    void *vptr = 0;
    int res = SWIG_ConvertPtr(argv[0], &vptr, SWIGTYPE_p_ANNMOD, 0);
    _v = SWIG_CheckState(res);
    if (_v) {
      return _wrap_ANNMOD_dGetStatDK__SWIG_0(self, args);
    }
  }
  if (argc == 2) {
    int _v;
    void *vptr = 0;
```

```
    int res = SWIG_ConvertPtr(argv[0], &vptr, SWIGTYPE_p_ANNMOD, 0);
    _v = SWIG_CheckState(res);
    if (_v) {
      {
        int res = SWIG_AsVal_long(argv[1], NULL);
        _v = SWIG_CheckState(res);
      }
      if (_v) {
        return _wrap_ANNMOD_dGetStatDK__SWIG_1(self, args);
      }
    }
  }

fail:
  SWIG_SetErrorMsg(PyExc_NotImplementedError,"Wrong number of arguments for overloaded f
Possible C/C++ prototypes are:\n    dGetStatDK()\n    dGetStatDK(long)\n");
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dGetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:ANNMOD_dGetQx",&obj0,&obj1,&obj2,&obj3,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dGetQx" "', argument
  }
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
```

```
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dGetQx" "', argumer
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dGetQx" "', argumer
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "ANNMOD_dGetQx" "', argumer
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "ANNMOD_dGetQx" "', argumer
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "ANNMOD_dGetQx" "', argumer
  }
  arg6 = static_cast< long >(val6);
  result = (double)(arg1)->dGetQx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_ANNMOD_dSetRelativeQxForTime(PyObject *SWIGUNUSEDPARM(self),
  PyObject *resultobj = 0;
  ANNMOD *arg1 = (ANNMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:ANNMOD_dSetRelativeQxForTime",&obj0,&obj1,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_ANNMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "ANNMOD_dSetRelativeQxForTime
  }
```

```
  arg1 = reinterpret_cast< ANNMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "ANNMOD_dSetRelativeQxForTi
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "ANNMOD_dSetRelativeQxForTi
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetRelativeQxForTime(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *ANNMOD_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_ANNMOD, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

SWIGINTERN PyObject *_wrap_new_VAMOD(PyObject *SWIGUNUSEDPARM(self), PyObject *args) {
  PyObject *resultobj = 0;
  VAMOD *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_VAMOD")) SWIG_fail;
  result = (VAMOD *)new VAMOD();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_VAMOD, SWIG_POINTER
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_VAMOD(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_VAMOD",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_VAMOD" "', argument '
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
```

```
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:VAMOD_dSetQx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetQx" "', argument '
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetQx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dSetQx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
```

```
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_dSetQx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_dSetQx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_dSetQx" "', argument
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetQx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetFx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:VAMOD_dSetFx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetFx" "', argument '
```

```
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetFx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dSetFx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_dSetFx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_dSetFx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_dSetFx" "', argument
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetFx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetSx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
```

```
  double val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:VAMOD_dSetSx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetSx" "', argument "
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetSx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dSetSx" "', argument
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_dSetSx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_dSetSx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_dSetSx" "', argument
  }
  arg6 = static_cast< double >(val6);
  result = (double)(arg1)->dSetSx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetBaseYear(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
```

```
  long arg5 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOO:VAMOD_dSetBaseYear",&obj0,&obj1,&obj2,&obj3,
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetBaseYear" "', argu
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetBaseYear" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dSetBaseYear" "', ar
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_dSetBaseYear" "', ar
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_dSetBaseYear" "', ar
  }
  arg5 = static_cast< long >(val5);
  result = (double)(arg1)->dSetBaseYear(arg2,arg3,arg4,arg5);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetActualYear(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
```

```
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:VAMOD_dSetActualYear",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetActualYear" "', ar
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetActualYear" "',
  }
  arg2 = static_cast< long >(val2);
  result = (double)(arg1)->dSetActualYear(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetDisc(PyObject *SWIGUNUSEDPARM(self), PyObject *args
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dSetDisc",&obj0,&obj1,&obj2)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetDisc" "', argument
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetDisc" "', argume
```

```
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dSetDisc" "', argume
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetDisc(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_iAnalyseToken(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  char *arg2 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:VAMOD_iAnalyseToken",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_iAnalyseToken" "', arg
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "VAMOD_iAnalyseToken" "', arg
  }
  arg2 = reinterpret_cast< char * >(buf2);
  result = (int)(arg1)->iAnalyseToken(arg2);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vGenerateTrajectory(PyObject *SWIGUNUSEDPARM(self), PyO
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:VAMOD_vGenerateTrajectory",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vGenerateTrajectory" '
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  (arg1)->vGenerateTrajectory();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanCF(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanCF",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanCF" "', argume
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanCF" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanCF" "', argu
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanCF(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanCFAnn(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanCFAnn",&obj0,&obj1,&obj2)) SWIG_
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanCFAnn" "', arg
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanCFAnn" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanCFAnn" "', a
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanCFAnn(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanCFPrem(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
```

```
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanCFPrem",&obj0,&obj1,&obj2)) SWIG
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanCFPrem" "', ar
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanCFPrem" "',
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanCFPrem" "',
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanCFPrem(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanCFMort(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanCFMort",&obj0,&obj1,&obj2)) SWIG
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanCFMort" "', ar
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanCFMort" "',
  }
```

```
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanCFMort" "',
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanCFMort(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanDK(PyObject *SWIGUNUSEDPARM(self), PyObject *ar
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanDK",&obj0,&obj1,&obj2)) SWIG_fai
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanDK" "', argume
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanDK" "', argu
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanDK" "', argu
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanDK(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}
```

```
SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanDKAnnMort(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanDKAnnMort",&obj0,&obj1,&obj2)) S
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanDKAnnMort" "',
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanDKAnnMort" '
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanDKAnnMort" '
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanDKAnnMort(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetMeanDKPrem(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
```

```
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetMeanDKPrem",&obj0,&obj1,&obj2)) SWIG
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetMeanDKPrem" "', ar
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetMeanDKPrem" "',
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetMeanDKPrem" "',
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetMeanDKPrem(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetDKDetail(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dGetDKDetail",&obj0,&obj1,&obj2)) SWIG_f
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetDKDetail" "', argu
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetDKDetail" "', ar
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
```

```
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetDKDetail" "', ar
  }
  arg3 = static_cast< long >(val3);
  result = (double)(arg1)->dGetDKDetail(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vNewSeed(PyObject *SWIGUNUSEDPARM(self), PyObject *args
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:VAMOD_vNewSeed",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vNewSeed" "', argument
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_vNewSeed" "', argume
  }
  arg2 = static_cast< long >(val2);
  (arg1)->vNewSeed(arg2);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vResetMeanResults(PyObject *SWIGUNUSEDPARM(self), PyObj
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:VAMOD_vResetMeanResults",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vResetMeanResults" "',
  }
```

```
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  (arg1)->vResetMeanResults();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_lSeed_set(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:VAMOD_lSeed_set",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_lSeed_set" "', argumer
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_lSeed_set" "', argum
  }
  arg2 = static_cast< long >(val2);
  if (arg1) (arg1)->lSeed = arg2;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_lSeed_get(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:VAMOD_lSeed_get",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_lSeed_get" "', argumer
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
```

```
  result = (long) ((arg1)->lSeed);
  resultobj = SWIG_From_long(static_cast< long >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vAddDeath(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOO:VAMOD_vAddDeath",&obj0,&obj1,&obj2,&obj3,
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vAddDeath" "', argumer
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg2 = static_cast< long >(val2);
```

```
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "VAMOD_vAddDeath" "', argum
  }
  arg8 = static_cast< double >(val8);
  (arg1)->vAddDeath(arg2,arg3,arg4,arg5,arg6,arg7,arg8);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vAddEndowment(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
```

```
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOO:VAMOD_vAddEndowment",&obj0,&obj1,&obj2,&o
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vAddEndowment" "', arg
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_vAddEndowment" "', a
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_vAddEndowment" "', a
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_vAddEndowment" "', a
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_vAddEndowment" "', a
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_vAddEndowment" "', a
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "VAMOD_vAddEndowment" "', a
  }
```

```
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "VAMOD_vAddEndowment" "', a
  }
  arg8 = static_cast< double >(val8);
  (arg1)->vAddEndowment(arg2,arg3,arg4,arg5,arg6,arg7,arg8);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vAddPremium(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  double arg6 ;
  double arg7 ;
  double arg8 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  double val6 ;
  int ecode6 = 0 ;
  double val7 ;
  int ecode7 = 0 ;
  double val8 ;
  int ecode8 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;
  PyObject * obj6 = 0 ;
  PyObject * obj7 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOOOO:VAMOD_vAddPremium",&obj0,&obj1,&obj2,&obj
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vAddPremium" "', argum
  }
```

```
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_double(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg6 = static_cast< double >(val6);
  ecode7 = SWIG_AsVal_double(obj6, &val7);
  if (!SWIG_IsOK(ecode7)) {
    SWIG_exception_fail(SWIG_ArgError(ecode7), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg7 = static_cast< double >(val7);
  ecode8 = SWIG_AsVal_double(obj7, &val8);
  if (!SWIG_IsOK(ecode8)) {
    SWIG_exception_fail(SWIG_ArgError(ecode8), "in method '" "VAMOD_vAddPremium" "', arg
  }
  arg8 = static_cast< double >(val8);
  (arg1)->vAddPremium(arg2,arg3,arg4,arg5,arg6,arg7,arg8);
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vUpdateOperator(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:VAMOD_vUpdateOperator",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
```

```
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vUpdateOperator" "', a
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  (arg1)->vUpdateOperator();
  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dGetQx(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  long arg3 ;
  long arg4 ;
  long arg5 ;
  long arg6 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  long val3 ;
  int ecode3 = 0 ;
  long val4 ;
  int ecode4 = 0 ;
  long val5 ;
  int ecode5 = 0 ;
  long val6 ;
  int ecode6 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;
  PyObject * obj4 = 0 ;
  PyObject * obj5 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOOOO:VAMOD_dGetQx",&obj0,&obj1,&obj2,&obj3,&obj4
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dGetQx" "', argument '
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dGetQx" "', argument
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_long(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dGetQx" "', argument
  }
```

```
  arg3 = static_cast< long >(val3);
  ecode4 = SWIG_AsVal_long(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_dGetQx" "', argument
  }
  arg4 = static_cast< long >(val4);
  ecode5 = SWIG_AsVal_long(obj4, &val5);
  if (!SWIG_IsOK(ecode5)) {
    SWIG_exception_fail(SWIG_ArgError(ecode5), "in method '" "VAMOD_dGetQx" "', argument
  }
  arg5 = static_cast< long >(val5);
  ecode6 = SWIG_AsVal_long(obj5, &val6);
  if (!SWIG_IsOK(ecode6)) {
    SWIG_exception_fail(SWIG_ArgError(ecode6), "in method '" "VAMOD_dGetQx" "', argument
  }
  arg6 = static_cast< long >(val6);
  result = (double)(arg1)->dGetQx(arg2,arg3,arg4,arg5,arg6);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_dSetRelativeQxForTime(PyObject *SWIGUNUSEDPARM(self), P
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  long arg2 ;
  double arg3 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  long val2 ;
  int ecode2 = 0 ;
  double val3 ;
  int ecode3 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOO:VAMOD_dSetRelativeQxForTime",&obj0,&obj1,&obj2
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_dSetRelativeQxForTime'
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_long(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_dSetRelativeQxForTim
  }
  arg2 = static_cast< long >(val2);
  ecode3 = SWIG_AsVal_double(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_dSetRelativeQxForTim
```

```
  }
  arg3 = static_cast< double >(val3);
  result = (double)(arg1)->dSetRelativeQxForTime(arg2,arg3);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_iReadInforce(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  int arg2 ;
  int arg3 ;
  char *arg4 = (char *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int val2 ;
  int ecode2 = 0 ;
  int val3 ;
  int ecode3 = 0 ;
  int res4 ;
  char *buf4 = 0 ;
  int alloc4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:VAMOD_iReadInforce",&obj0,&obj1,&obj2,&obj3))
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_iReadInforce" "', argu
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_int(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_iReadInforce" "', ar
  }
  arg2 = static_cast< int >(val2);
  ecode3 = SWIG_AsVal_int(obj2, &val3);
  if (!SWIG_IsOK(ecode3)) {
    SWIG_exception_fail(SWIG_ArgError(ecode3), "in method '" "VAMOD_iReadInforce" "', ar
  }
  arg3 = static_cast< int >(val3);
  res4 = SWIG_AsCharPtrAndSize(obj3, &buf4, NULL, &alloc4);
  if (!SWIG_IsOK(res4)) {
    SWIG_exception_fail(SWIG_ArgError(res4), "in method '" "VAMOD_iReadInforce" "', argu
  }
  arg4 = reinterpret_cast< char * >(buf4);
  result = (int)(arg1)->iReadInforce(arg2,arg3,arg4);
  resultobj = SWIG_From_int(static_cast< int >(result));
```

```
  if (alloc4 == SWIG_NEWOBJ) delete[] buf4;
  return resultobj;
fail:
  if (alloc4 == SWIG_NEWOBJ) delete[] buf4;
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_vPrintTex(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  char *arg2 = (char *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:VAMOD_vPrintTex",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_vPrintTex" "', argumer
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "VAMOD_vPrintTex" "', argumer
  }
  arg2 = reinterpret_cast< char * >(buf2);
  (arg1)->vPrintTex(arg2);
  resultobj = SWIG_Py_Void();
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_symPara_set(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  VAPAR arg2 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  void *argp2 ;
  int res2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:VAMOD_symPara_set",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
```

```
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_symPara_set" "', argum
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  {
    res2 = SWIG_ConvertPtr(obj1, &argp2, SWIGTYPE_p_VAPAR,  0  | 0);
    if (!SWIG_IsOK(res2)) {
      SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "VAMOD_symPara_set" "', arg
    }
    if (!argp2) {
      SWIG_exception_fail(SWIG_ValueError, "invalid null reference " "in method '" "VAMO
    } else {
      VAPAR * temp = reinterpret_cast< VAPAR * >(argp2);
      arg2 = *temp;
      if (SWIG_IsNewObj(res2)) delete temp;
    }
  }
  if (arg1) (arg1)->symPara = arg2;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_symPara_get(PyObject *SWIGUNUSEDPARM(self), PyObject *a
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  VAPAR result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:VAMOD_symPara_get",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_symPara_get" "', argum
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  result =  ((arg1)->symPara);
  resultobj = SWIG_NewPointerObj((new VAPAR(static_cast< const VAPAR& >(result))), SWIGT
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_VAMOD_iSetTable(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  VAMOD *arg1 = (VAMOD *) 0 ;
  int arg2 ;
  char *arg3 = (char *) 0 ;
```

```
  double arg4 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int val2 ;
  int ecode2 = 0 ;
  int res3 ;
  char *buf3 = 0 ;
  int alloc3 = 0 ;
  double val4 ;
  int ecode4 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;
  PyObject * obj2 = 0 ;
  PyObject * obj3 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OOOO:VAMOD_iSetTable",&obj0,&obj1,&obj2,&obj3)) SW
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_VAMOD, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "VAMOD_iSetTable" "', argumer
  }
  arg1 = reinterpret_cast< VAMOD * >(argp1);
  ecode2 = SWIG_AsVal_int(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "VAMOD_iSetTable" "', argum
  }
  arg2 = static_cast< int >(val2);
  res3 = SWIG_AsCharPtrAndSize(obj2, &buf3, NULL, &alloc3);
  if (!SWIG_IsOK(res3)) {
    SWIG_exception_fail(SWIG_ArgError(res3), "in method '" "VAMOD_iSetTable" "', argumer
  }
  arg3 = reinterpret_cast< char * >(buf3);
  ecode4 = SWIG_AsVal_double(obj3, &val4);
  if (!SWIG_IsOK(ecode4)) {
    SWIG_exception_fail(SWIG_ArgError(ecode4), "in method '" "VAMOD_iSetTable" "', argum
  }
  arg4 = static_cast< double >(val4);
  result = (int)(arg1)->iSetTable(arg2,arg3,arg4);
  resultobj = SWIG_From_int(static_cast< int >(result));
  if (alloc3 == SWIG_NEWOBJ) delete[] buf3;
  return resultobj;
fail:
  if (alloc3 == SWIG_NEWOBJ) delete[] buf3;
  return NULL;
}


SWIGINTERN PyObject *VAMOD_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *args)
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_VAMOD, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}
```

```
SWIGINTERN PyObject *_wrap_new_TABLESERVER(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  TABLESERVER *result = 0 ;

  if (!PyArg_ParseTuple(args,(char *)":new_TABLESERVER")) SWIG_fail;
  result = (TABLESERVER *)new TABLESERVER();
  resultobj = SWIG_NewPointerObj(SWIG_as_voidptr(result), SWIGTYPE_p_TABLESERVER, SWIG_P
0 );
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_vSetTable(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  char *arg2 = (char *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  int res2 ;
  char *buf2 = 0 ;
  int alloc2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:TABLESERVER_vSetTable",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_vSetTable" "', a
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  res2 = SWIG_AsCharPtrAndSize(obj1, &buf2, NULL, &alloc2);
  if (!SWIG_IsOK(res2)) {
    SWIG_exception_fail(SWIG_ArgError(res2), "in method '" "TABLESERVER_vSetTable" "', a
  }
  arg2 = reinterpret_cast< char * >(buf2);
  (arg1)->vSetTable(arg2);
  resultobj = SWIG_Py_Void();
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return resultobj;
fail:
  if (alloc2 == SWIG_NEWOBJ) delete[] buf2;
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_dGetValue(PyObject *SWIGUNUSEDPARM(self), PyObjec
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  int arg2 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  int val2 ;
  int ecode2 = 0 ;
  PyObject * obj0 = 0 ;
  PyObject * obj1 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"OO:TABLESERVER_dGetValue",&obj0,&obj1)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_dGetValue" "', a
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  ecode2 = SWIG_AsVal_int(obj1, &val2);
  if (!SWIG_IsOK(ecode2)) {
    SWIG_exception_fail(SWIG_ArgError(ecode2), "in method '" "TABLESERVER_dGetValue" "',
  }
  arg2 = static_cast< int >(val2);
  result = (double)(arg1)->dGetValue(arg2);
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_iTableNumber(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_iTableNumber",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_iTableNumber" "'
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (int)(arg1)->iTableNumber();
  resultobj = SWIG_From_int(static_cast< int >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_iX0(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;
```

```
  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_iX0",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_iX0" "', argumer
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (int)(arg1)->iX0();
  resultobj = SWIG_From_int(static_cast< int >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_iXOmega(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_iXOmega",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_iXOmega" "', arg
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (int)(arg1)->iXOmega();
  resultobj = SWIG_From_int(static_cast< int >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_iT0(PyObject *SWIGUNUSEDPARM(self), PyObject *arg
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_iT0",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_iT0" "', argumer
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (int)(arg1)->iT0();
  resultobj = SWIG_From_int(static_cast< int >(result));
  return resultobj;
fail:
```

```
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_dITech(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  double result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_dITech",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_dITech" "', argu
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (double)(arg1)->dITech();
  resultobj = SWIG_From_double(static_cast< double >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_iGender(PyObject *SWIGUNUSEDPARM(self), PyObject
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  int result;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_iGender",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_iGender" "', arg
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (int)(arg1)->iGender();
  resultobj = SWIG_From_int(static_cast< int >(result));
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_TABLESERVER_pcAllTarifs(PyObject *SWIGUNUSEDPARM(self), PyOb
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  char *result = 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
```

```
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:TABLESERVER_pcAllTarifs",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, 0 |  0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "TABLESERVER_pcAllTarifs" "',
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  result = (char *)(arg1)->pcAllTarifs();
  resultobj = SWIG_FromCharPtr((const char *)result);
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *_wrap_delete_TABLESERVER(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *resultobj = 0;
  TABLESERVER *arg1 = (TABLESERVER *) 0 ;
  void *argp1 = 0 ;
  int res1 = 0 ;
  PyObject * obj0 = 0 ;

  if (!PyArg_ParseTuple(args,(char *)"O:delete_TABLESERVER",&obj0)) SWIG_fail;
  res1 = SWIG_ConvertPtr(obj0, &argp1,SWIGTYPE_p_TABLESERVER, SWIG_POINTER_DISOWN |
0 );
  if (!SWIG_IsOK(res1)) {
    SWIG_exception_fail(SWIG_ArgError(res1), "in method '" "delete_TABLESERVER" "', argu
  }
  arg1 = reinterpret_cast< TABLESERVER * >(argp1);
  delete arg1;

  resultobj = SWIG_Py_Void();
  return resultobj;
fail:
  return NULL;
}


SWIGINTERN PyObject *TABLESERVER_swigregister(PyObject *SWIGUNUSEDPARM(self), PyObject *
  PyObject *obj;
  if (!PyArg_ParseTuple(args,(char*)"O|swigregister", &obj)) return NULL;
  SWIG_TypeNewClientData(SWIGTYPE_p_TABLESERVER, SWIG_NewClientData(obj));
  return SWIG_Py_Void();
}

static PyMethodDef SwigMethods[] = {
         { (char *)"new_MARKOVLV", _wrap_new_MARKOVLV, METH_VARARGS, NULL},
         { (char *)"delete_MARKOVLV", _wrap_delete_MARKOVLV, METH_VARARGS, NULL},
         { (char *)"MARKOVLV_vReset", _wrap_MARKOVLV_vReset, METH_VARARGS, NULL},
         { (char *)"MARKOVLV_vSetInternals", _wrap_MARKOVLV_vSetInternals, METH_VARARGS,
         { (char *)"MARKOVLV_vSetStartTime", _wrap_MARKOVLV_vSetStartTime, METH_VARARGS,
         { (char *)"MARKOVLV_vSetStopTime", _wrap_MARKOVLV_vSetStopTime, METH_VARARGS, N
         { (char *)"MARKOVLV_vSetNrStates", _wrap_MARKOVLV_vSetNrStates, METH_VARARGS, N
```

```
{ (char *)"MARKOVLV_vSetGetData", _wrap_MARKOVLV_vSetGetData, METH_VARARGS, NUI
{ (char *)"MARKOVLV_dSetPre", _wrap_MARKOVLV_dSetPre, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dSetPost", _wrap_MARKOVLV_dSetPost, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dSetPij", _wrap_MARKOVLV_dSetPij, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dSetDisc", _wrap_MARKOVLV_dSetDisc, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_vSetInterestModel", _wrap_MARKOVLV_vSetInterestModel, METH_
{ (char *)"MARKOVLV_vSetDefaultNrMoments", _wrap_MARKOVLV_vSetDefaultNrMoments,
{ (char *)"MARKOVLV_dGetDK", _wrap_MARKOVLV_dGetDK, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dGetCF", _wrap_MARKOVLV_dGetCF, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dGetRP", _wrap_MARKOVLV_dGetRP, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dGetSP", _wrap_MARKOVLV_dGetSP, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dGetRegP", _wrap_MARKOVLV_dGetRegP, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_lSetFolgezustand", _wrap_MARKOVLV_lSetFolgezustand, METH_VA
{ (char *)"MARKOVLV_lGetMaxTime", _wrap_MARKOVLV_lGetMaxTime, METH_VARARGS, NUI
{ (char *)"MARKOVLV_lGetNrStates", _wrap_MARKOVLV_lGetNrStates, METH_VARARGS, N
{ (char *)"MARKOVLV_lGetStartTime", _wrap_MARKOVLV_lGetStartTime, METH_VARARGS,
{ (char *)"MARKOVLV_lGetStopTime", _wrap_MARKOVLV_lGetStopTime, METH_VARARGS, N
{ (char *)"MARKOVLV_dAddBenefits_set", _wrap_MARKOVLV_dAddBenefits_set, METH_VA
{ (char *)"MARKOVLV_dAddBenefits_get", _wrap_MARKOVLV_dAddBenefits_get, METH_VA
{ (char *)"MARKOVLV_vSetInitState", _wrap_MARKOVLV_vSetInitState, METH_VARARGS,
{ (char *)"MARKOVLV_vGenerateTrajectory", _wrap_MARKOVLV_vGenerateTrajectory, N
{ (char *)"MARKOVLV_vGetState", _wrap_MARKOVLV_vGetState, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_dGetRandCF", _wrap_MARKOVLV_dGetRandCF, METH_VARARGS, NULL}
{ (char *)"MARKOVLV_dGetRandDK", _wrap_MARKOVLV_dGetRandDK, METH_VARARGS, NULL}
{ (char *)"MARKOVLV_dGetMeanCF", _wrap_MARKOVLV_dGetMeanCF, METH_VARARGS, NULL}
{ (char *)"MARKOVLV_dGetMeanDK", _wrap_MARKOVLV_dGetMeanDK, METH_VARARGS, NULL}
{ (char *)"MARKOVLV_vNewSeed", _wrap_MARKOVLV_vNewSeed, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_vResetMeanResults", _wrap_MARKOVLV_vResetMeanResults, METH_
{ (char *)"MARKOVLV_lSeed_set", _wrap_MARKOVLV_lSeed_set, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_lSeed_get", _wrap_MARKOVLV_lSeed_get, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_vPrintTeX", _wrap_MARKOVLV_vPrintTeX, METH_VARARGS, NULL},
{ (char *)"MARKOVLV_swigregister", MARKOVLV_swigregister, METH_VARARGS, NULL},
{ (char *)"new_CAPITALLV", _wrap_new_CAPITALLV, METH_VARARGS, NULL},
{ (char *)"delete_CAPITALLV", _wrap_delete_CAPITALLV, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_iSetTable", _wrap_CAPITALLV_iSetTable, METH_VARARGS, NULL}
{ (char *)"CAPITALLV_vSetStartTime", _wrap_CAPITALLV_vSetStartTime, METH_VARARG
{ (char *)"CAPITALLV_vSetStopTime", _wrap_CAPITALLV_vSetStopTime, METH_VARARGS,
{ (char *)"CAPITALLV_vSetSurvival", _wrap_CAPITALLV_vSetSurvival, METH_VARARGS,
{ (char *)"CAPITALLV_vSetDeath", _wrap_CAPITALLV_vSetDeath, METH_VARARGS, NULL}
{ (char *)"CAPITALLV_vSetPremium", _wrap_CAPITALLV_vSetPremium, METH_VARARGS, N
{ (char *)"CAPITALLV_vSetSurvivalGen", _wrap_CAPITALLV_vSetSurvivalGen, METH_VA
{ (char *)"CAPITALLV_vSetDeathGen", _wrap_CAPITALLV_vSetDeathGen, METH_VARARGS,
{ (char *)"CAPITALLV_vSetPremiumGen", _wrap_CAPITALLV_vSetPremiumGen, METH_VARA
{ (char *)"CAPITALLV_dSetQx", _wrap_CAPITALLV_dSetQx, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dSetFx", _wrap_CAPITALLV_dSetFx, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dSetBaseYear", _wrap_CAPITALLV_dSetBaseYear, METH_VARARGS,
{ (char *)"CAPITALLV_dSetActualYear", _wrap_CAPITALLV_dSetActualYear, METH_VARA
{ (char *)"CAPITALLV_dSetDisc", _wrap_CAPITALLV_dSetDisc, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dGetDK", _wrap_CAPITALLV_dGetDK, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dGetCF", _wrap_CAPITALLV_dGetCF, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dGetQx", _wrap_CAPITALLV_dGetQx, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dSetQx2Level", _wrap_CAPITALLV_dSetQx2Level, METH_VARARGS,
{ (char *)"CAPITALLV_dSetSx2", _wrap_CAPITALLV_dSetSx2, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_dSetRDR", _wrap_CAPITALLV_dSetRDR, METH_VARARGS, NULL},
```

```
{ (char *)"CAPITALLV_dSetSurenderPenaltyInMR", _wrap_CAPITALLV_dSetSurenderPena
{ (char *)"CAPITALLV_dSetSHMarginInMR", _wrap_CAPITALLV_dSetSHMarginInMR, METH_
{ (char *)"CAPITALLV_dSetSolaCapitalInMR", _wrap_CAPITALLV_dSetSolaCapitalInMR,
{ (char *)"CAPITALLV_dSetInvReturn", _wrap_CAPITALLV_dSetInvReturn, METH_VARARG
{ (char *)"CAPITALLV_dGetEV", _wrap_CAPITALLV_dGetEV, METH_VARARGS, NULL},
{ (char *)"CAPITALLV_swigregister", CAPITALLV_swigregister, METH_VARARGS, NULL}
{ (char *)"new_ANNUITYLV", _wrap_new_ANNUITYLV, METH_VARARGS, NULL},
{ (char *)"delete_ANNUITYLV", _wrap_delete_ANNUITYLV, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_iSetTable", _wrap_ANNUITYLV_iSetTable, METH_VARARGS, NULL}
{ (char *)"ANNUITYLV_vSetStartTime", _wrap_ANNUITYLV_vSetStartTime, METH_VARARG
{ (char *)"ANNUITYLV_vSetStopTime", _wrap_ANNUITYLV_vSetStopTime, METH_VARARGS,
{ (char *)"ANNUITYLV_vSetSAge", _wrap_ANNUITYLV_vSetSAge, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_vSetG", _wrap_ANNUITYLV_vSetG, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_vSetMaxProj", _wrap_ANNUITYLV_vSetMaxProj, METH_VARARGS, N
{ (char *)"ANNUITYLV_dSetQx", _wrap_ANNUITYLV_dSetQx, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dSetFx", _wrap_ANNUITYLV_dSetFx, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dSetSx", _wrap_ANNUITYLV_dSetSx, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dSetBaseYear", _wrap_ANNUITYLV_dSetBaseYear, METH_VARARGS,
{ (char *)"ANNUITYLV_dSetActualYear", _wrap_ANNUITYLV_dSetActualYear, METH_VARA
{ (char *)"ANNUITYLV_dSetDisc", _wrap_ANNUITYLV_dSetDisc, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dGetDK", _wrap_ANNUITYLV_dGetDK, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dGetCF", _wrap_ANNUITYLV_dGetCF, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dGetQx", _wrap_ANNUITYLV_dGetQx, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dGetTqx", _wrap_ANNUITYLV_dGetTqx, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dGetTpx", _wrap_ANNUITYLV_dGetTpx, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dSetPre", _wrap_ANNUITYLV_dSetPre, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV_dSetRelativeQxForTime", _wrap_ANNUITYLV_dSetRelativeQxForT
{ (char *)"ANNUITYLV_vLeistReset", _wrap_ANNUITYLV_vLeistReset, METH_VARARGS, N
{ (char *)"ANNUITYLV_vSetLeistLinear", _wrap_ANNUITYLV_vSetLeistLinear, METH_VA
{ (char *)"ANNUITYLV_vSetLeistExp", _wrap_ANNUITYLV_vSetLeistExp, METH_VARARGS,
{ (char *)"ANNUITYLV_swigregister", ANNUITYLV_swigregister, METH_VARARGS, NULL}
{ (char *)"new_ANNUITYLV2", _wrap_new_ANNUITYLV2, METH_VARARGS, NULL},
{ (char *)"delete_ANNUITYLV2", _wrap_delete_ANNUITYLV2, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_iSetTable1", _wrap_ANNUITYLV2_iSetTable1, METH_VARARGS, N
{ (char *)"ANNUITYLV2_iSetTable2", _wrap_ANNUITYLV2_iSetTable2, METH_VARARGS, N
{ (char *)"ANNUITYLV2_vSetStartTime", _wrap_ANNUITYLV2_vSetStartTime, METH_VARA
{ (char *)"ANNUITYLV2_vSetStopTime", _wrap_ANNUITYLV2_vSetStopTime, METH_VARARG
{ (char *)"ANNUITYLV2_vSetSAge1", _wrap_ANNUITYLV2_vSetSAge1, METH_VARARGS, NUI
{ (char *)"ANNUITYLV2_vSetSAge2", _wrap_ANNUITYLV2_vSetSAge2, METH_VARARGS, NUI
{ (char *)"ANNUITYLV2_dSetQx1", _wrap_ANNUITYLV2_dSetQx1, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dSetFx1", _wrap_ANNUITYLV2_dSetFx1, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dSetQx2", _wrap_ANNUITYLV2_dSetQx2, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dSetFx2", _wrap_ANNUITYLV2_dSetFx2, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dSetBaseYear", _wrap_ANNUITYLV2_dSetBaseYear, METH_VARARG
{ (char *)"ANNUITYLV2_dSetActualYear", _wrap_ANNUITYLV2_dSetActualYear, METH_VA
{ (char *)"ANNUITYLV2_dSetDisc", _wrap_ANNUITYLV2_dSetDisc, METH_VARARGS, NULL}
{ (char *)"ANNUITYLV2_dGetDK", _wrap_ANNUITYLV2_dGetDK, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dGetCF", _wrap_ANNUITYLV2_dGetCF, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dGetQx1", _wrap_ANNUITYLV2_dGetQx1, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dGetQx2", _wrap_ANNUITYLV2_dGetQx2, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_dSetY_Minus_X", _wrap_ANNUITYLV2_dSetY_Minus_X, METH_VARA
{ (char *)"ANNUITYLV2_dSetBenefit", _wrap_ANNUITYLV2_dSetBenefit, METH_VARARGS,
{ (char *)"ANNUITYLV2_dSetPre", _wrap_ANNUITYLV2_dSetPre, METH_VARARGS, NULL},
{ (char *)"ANNUITYLV2_vLeistReset", _wrap_ANNUITYLV2_vLeistReset, METH_VARARGS,
```

```
{ (char *)"ANNUITYLV2_vSetLeistLinear", _wrap_ANNUITYLV2_vSetLeistLinear, METH_
{ (char *)"ANNUITYLV2_vSetLeistExp", _wrap_ANNUITYLV2_vSetLeistExp, METH_VARARG
{ (char *)"ANNUITYLV2_swigregister", ANNUITYLV2_swigregister, METH_VARARGS, NUL
{ (char *)"new_WIDDOWLV", _wrap_new_WIDDOWLV, METH_VARARGS, NULL},
{ (char *)"delete_WIDDOWLV", _wrap_delete_WIDDOWLV, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_vSetStartTime", _wrap_WIDDOWLV_vSetStartTime, METH_VARARGS,
{ (char *)"WIDDOWLV_vSetStopTime", _wrap_WIDDOWLV_vSetStopTime, METH_VARARGS, N
{ (char *)"WIDDOWLV_dSetQx", _wrap_WIDDOWLV_dSetQx, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetQy", _wrap_WIDDOWLV_dSetQy, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetFx", _wrap_WIDDOWLV_dSetFx, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetFy", _wrap_WIDDOWLV_dSetFy, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetHx", _wrap_WIDDOWLV_dSetHx, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetYx", _wrap_WIDDOWLV_dSetYx, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetBaseYear", _wrap_WIDDOWLV_dSetBaseYear, METH_VARARGS, N
{ (char *)"WIDDOWLV_dSetActualYear", _wrap_WIDDOWLV_dSetActualYear, METH_VARARG
{ (char *)"WIDDOWLV_dSetDisc", _wrap_WIDDOWLV_dSetDisc, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dGetDK", _wrap_WIDDOWLV_dGetDK, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dGetCF", _wrap_WIDDOWLV_dGetCF, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dGetQx", _wrap_WIDDOWLV_dGetQx, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_dSetPre", _wrap_WIDDOWLV_dSetPre, METH_VARARGS, NULL},
{ (char *)"WIDDOWLV_vLeistReset", _wrap_WIDDOWLV_vLeistReset, METH_VARARGS, NUL
{ (char *)"WIDDOWLV_vSetLeistLinear", _wrap_WIDDOWLV_vSetLeistLinear, METH_VARA
{ (char *)"WIDDOWLV_vSetLeistExp", _wrap_WIDDOWLV_vSetLeistExp, METH_VARARGS, N
{ (char *)"WIDDOWLV_swigregister", WIDDOWLV_swigregister, METH_VARARGS, NULL},
{ (char *)"new_GLMOD", _wrap_new_GLMOD, METH_VARARGS, NULL},
{ (char *)"delete_GLMOD", _wrap_delete_GLMOD, METH_VARARGS, NULL},
{ (char *)"GLMOD_dSetQx", _wrap_GLMOD_dSetQx, METH_VARARGS, NULL},
{ (char *)"GLMOD_dSetFx", _wrap_GLMOD_dSetFx, METH_VARARGS, NULL},
{ (char *)"GLMOD_dSetSx", _wrap_GLMOD_dSetSx, METH_VARARGS, NULL},
{ (char *)"GLMOD_dSetBaseYear", _wrap_GLMOD_dSetBaseYear, METH_VARARGS, NULL},
{ (char *)"GLMOD_dSetActualYear", _wrap_GLMOD_dSetActualYear, METH_VARARGS, NUL
{ (char *)"GLMOD_dSetDisc", _wrap_GLMOD_dSetDisc, METH_VARARGS, NULL},
{ (char *)"GLMOD_vStress", _wrap_GLMOD_vStress, METH_VARARGS, NULL},
{ (char *)"GLMOD_vAddAnnuity", _wrap_GLMOD_vAddAnnuity, METH_VARARGS, NULL},
{ (char *)"GLMOD_vAddEndowment", _wrap_GLMOD_vAddEndowment, METH_VARARGS, NULL}
{ (char *)"GLMOD_vAddWiddow", _wrap_GLMOD_vAddWiddow, METH_VARARGS, NULL},
{ (char *)"GLMOD_vSetRKWAnnuity", _wrap_GLMOD_vSetRKWAnnuity, METH_VARARGS, NUL
{ (char *)"GLMOD_vSetRKWEndowment", _wrap_GLMOD_vSetRKWEndowment, METH_VARARGS,
{ (char *)"GLMOD_vUpdateOperator", _wrap_GLMOD_vUpdateOperator, METH_VARARGS, N
{ (char *)"GLMOD_dGetDK", _wrap_GLMOD_dGetDK, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetDKDetail", _wrap_GLMOD_dGetDKDetail, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetDKTilde", _wrap_GLMOD_dGetDKTilde, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetFVDK", _wrap_GLMOD_dGetFVDK, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetCF", _wrap_GLMOD_dGetCF, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetCFDetail", _wrap_GLMOD_dGetCFDetail, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetStatDK", _wrap_GLMOD_dGetStatDK, METH_VARARGS, NULL},
{ (char *)"GLMOD_dGetQx", _wrap_GLMOD_dGetQx, METH_VARARGS, NULL},
{ (char *)"GLMOD_dSetRelativeQxForTime", _wrap_GLMOD_dSetRelativeQxForTime, MET
{ (char *)"GLMOD_iReadInforce", _wrap_GLMOD_iReadInforce, METH_VARARGS, NULL},
{ (char *)"GLMOD_vPrintTex", _wrap_GLMOD_vPrintTex, METH_VARARGS, NULL},
{ (char *)"GLMOD_swigregister", GLMOD_swigregister, METH_VARARGS, NULL},
{ (char *)"new_ANNMOD", _wrap_new_ANNMOD, METH_VARARGS, NULL},
{ (char *)"delete_ANNMOD", _wrap_delete_ANNMOD, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dSetQx", _wrap_ANNMOD_dSetQx, METH_VARARGS, NULL},
```

```
{ (char *)"ANNMOD_dSetFx", _wrap_ANNMOD_dSetFx, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dSetSx", _wrap_ANNMOD_dSetSx, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dSetBaseYear", _wrap_ANNMOD_dSetBaseYear, METH_VARARGS, NULL}
{ (char *)"ANNMOD_dSetActualYear", _wrap_ANNMOD_dSetActualYear, METH_VARARGS, N
{ (char *)"ANNMOD_dSetDisc", _wrap_ANNMOD_dSetDisc, METH_VARARGS, NULL},
{ (char *)"ANNMOD_vAddAnnuity1", _wrap_ANNMOD_vAddAnnuity1, METH_VARARGS, NULL}
{ (char *)"ANNMOD_vAddAnnuity0", _wrap_ANNMOD_vAddAnnuity0, METH_VARARGS, NULL}
{ (char *)"ANNMOD_vAddAnnuity2xy", _wrap_ANNMOD_vAddAnnuity2xy, METH_VARARGS, N
{ (char *)"ANNMOD_vAddAnnuity2xyBar", _wrap_ANNMOD_vAddAnnuity2xyBar, METH_VARA
{ (char *)"ANNMOD_vAddAnnuity2xToy", _wrap_ANNMOD_vAddAnnuity2xToy, METH_VARARG
{ (char *)"ANNMOD_vAddAnnuity2yTox", _wrap_ANNMOD_vAddAnnuity2yTox, METH_VARARG
{ (char *)"ANNMOD_vUpdateOperator", _wrap_ANNMOD_vUpdateOperator, METH_VARARGS,
{ (char *)"ANNMOD_dGetDK", _wrap_ANNMOD_dGetDK, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dGetFVDK", _wrap_ANNMOD_dGetFVDK, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dGetCF", _wrap_ANNMOD_dGetCF, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dGetStatDK", _wrap_ANNMOD_dGetStatDK, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dGetQx", _wrap_ANNMOD_dGetQx, METH_VARARGS, NULL},
{ (char *)"ANNMOD_dSetRelativeQxForTime", _wrap_ANNMOD_dSetRelativeQxForTime, N
{ (char *)"ANNMOD_swigregister", ANNMOD_swigregister, METH_VARARGS, NULL},
{ (char *)"new_VAMOD", _wrap_new_VAMOD, METH_VARARGS, NULL},
{ (char *)"delete_VAMOD", _wrap_delete_VAMOD, METH_VARARGS, NULL},
{ (char *)"VAMOD_dSetQx", _wrap_VAMOD_dSetQx, METH_VARARGS, NULL},
{ (char *)"VAMOD_dSetFx", _wrap_VAMOD_dSetFx, METH_VARARGS, NULL},
{ (char *)"VAMOD_dSetSx", _wrap_VAMOD_dSetSx, METH_VARARGS, NULL},
{ (char *)"VAMOD_dSetBaseYear", _wrap_VAMOD_dSetBaseYear, METH_VARARGS, NULL},
{ (char *)"VAMOD_dSetActualYear", _wrap_VAMOD_dSetActualYear, METH_VARARGS, NUI
{ (char *)"VAMOD_dSetDisc", _wrap_VAMOD_dSetDisc, METH_VARARGS, NULL},
{ (char *)"VAMOD_iAnalyseToken", _wrap_VAMOD_iAnalyseToken, METH_VARARGS, NULL}
{ (char *)"VAMOD_vGenerateTrajectory", _wrap_VAMOD_vGenerateTrajectory, METH_VA
{ (char *)"VAMOD_dGetMeanCF", _wrap_VAMOD_dGetMeanCF, METH_VARARGS, NULL},
{ (char *)"VAMOD_dGetMeanCFAnn", _wrap_VAMOD_dGetMeanCFAnn, METH_VARARGS, NULL}
{ (char *)"VAMOD_dGetMeanCFPrem", _wrap_VAMOD_dGetMeanCFPrem, METH_VARARGS, NUI
{ (char *)"VAMOD_dGetMeanCFMort", _wrap_VAMOD_dGetMeanCFMort, METH_VARARGS, NUI
{ (char *)"VAMOD_dGetMeanDK", _wrap_VAMOD_dGetMeanDK, METH_VARARGS, NULL},
{ (char *)"VAMOD_dGetMeanDKAnnMort", _wrap_VAMOD_dGetMeanDKAnnMort, METH_VARARG
{ (char *)"VAMOD_dGetMeanDKPrem", _wrap_VAMOD_dGetMeanDKPrem, METH_VARARGS, NUI
{ (char *)"VAMOD_dGetDKDetail", _wrap_VAMOD_dGetDKDetail, METH_VARARGS, NULL},
{ (char *)"VAMOD_vNewSeed", _wrap_VAMOD_vNewSeed, METH_VARARGS, NULL},
{ (char *)"VAMOD_vResetMeanResults", _wrap_VAMOD_vResetMeanResults, METH_VARARG
{ (char *)"VAMOD_lSeed_set", _wrap_VAMOD_lSeed_set, METH_VARARGS, NULL},
{ (char *)"VAMOD_lSeed_get", _wrap_VAMOD_lSeed_get, METH_VARARGS, NULL},
{ (char *)"VAMOD_vAddDeath", _wrap_VAMOD_vAddDeath, METH_VARARGS, NULL},
{ (char *)"VAMOD_vAddEndowment", _wrap_VAMOD_vAddEndowment, METH_VARARGS, NULL}
{ (char *)"VAMOD_vAddPremium", _wrap_VAMOD_vAddPremium, METH_VARARGS, NULL},
{ (char *)"VAMOD_vUpdateOperator", _wrap_VAMOD_vUpdateOperator, METH_VARARGS, N
{ (char *)"VAMOD_dGetQx", _wrap_VAMOD_dGetQx, METH_VARARGS, NULL},
{ (char *)"VAMOD_dSetRelativeQxForTime", _wrap_VAMOD_dSetRelativeQxForTime, MET
{ (char *)"VAMOD_iReadInforce", _wrap_VAMOD_iReadInforce, METH_VARARGS, NULL},
{ (char *)"VAMOD_vPrintTex", _wrap_VAMOD_vPrintTex, METH_VARARGS, NULL},
{ (char *)"VAMOD_symPara_set", _wrap_VAMOD_symPara_set, METH_VARARGS, NULL},
{ (char *)"VAMOD_symPara_get", _wrap_VAMOD_symPara_get, METH_VARARGS, NULL},
{ (char *)"VAMOD_iSetTable", _wrap_VAMOD_iSetTable, METH_VARARGS, NULL},
{ (char *)"VAMOD_swigregister", VAMOD_swigregister, METH_VARARGS, NULL},
{ (char *)"new_TABLESERVER", _wrap_new_TABLESERVER, METH_VARARGS, NULL},
```

```
        { (char *)"TABLESERVER_vSetTable", _wrap_TABLESERVER_vSetTable, METH_VARARGS, N
        { (char *)"TABLESERVER_dGetValue", _wrap_TABLESERVER_dGetValue, METH_VARARGS, N
        { (char *)"TABLESERVER_iTableNumber", _wrap_TABLESERVER_iTableNumber, METH_VARA
        { (char *)"TABLESERVER_iX0", _wrap_TABLESERVER_iX0, METH_VARARGS, NULL},
        { (char *)"TABLESERVER_iXOmega", _wrap_TABLESERVER_iXOmega, METH_VARARGS, NULL}
        { (char *)"TABLESERVER_iT0", _wrap_TABLESERVER_iT0, METH_VARARGS, NULL},
        { (char *)"TABLESERVER_dITech", _wrap_TABLESERVER_dITech, METH_VARARGS, NULL},
        { (char *)"TABLESERVER_iGender", _wrap_TABLESERVER_iGender, METH_VARARGS, NULL}
        { (char *)"TABLESERVER_pcAllTarifs", _wrap_TABLESERVER_pcAllTarifs, METH_VARARG
        { (char *)"delete_TABLESERVER", _wrap_delete_TABLESERVER, METH_VARARGS, NULL},
        { (char *)"TABLESERVER_swigregister", TABLESERVER_swigregister, METH_VARARGS, N
        { NULL, NULL, 0, NULL }
};


/* -------- TYPE CONVERSION AND EQUIVALENCE RULES (BEGIN) -------- */

static void *_p_WIDDOWLVTo_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((WIDDOWLV *) x));
}
static void *_p_CAPITALLVTo_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((CAPITALLV *) x));
}
static void *_p_GLMODTo_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((GLMOD *) x));
}
static void *_p_VAMODTo_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((VAMOD *) x));
}
static void *_p_ANNUITYLV2To_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((ANNUITYLV2 *) x));
}
static void *_p_ANNUITYLVTo_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((ANNUITYLV *) x));
}
static void *_p_ANNMODTo_p_MARKOVLV(void *x) {
    return (void *)((MARKOVLV *)  ((ANNMOD *) x));
}
static swig_type_info _swigt__p_ANNMOD = {"_p_ANNMOD", "ANNMOD *", 0, 0, (void*)0, 0};
static swig_type_info _swigt__p_ANNUITYLV = {"_p_ANNUITYLV", "ANNUITYLV *", 0, 0, (void*
static swig_type_info _swigt__p_ANNUITYLV2 = {"_p_ANNUITYLV2", "ANNUITYLV2 *", 0, 0, (vo
static swig_type_info _swigt__p_CAPITALLV = {"_p_CAPITALLV", "CAPITALLV *", 0, 0, (void*
static swig_type_info _swigt__p_FILE = {"_p_FILE", "FILE *", 0, 0, (void*)0, 0};
static swig_type_info _swigt__p_GLMOD = {"_p_GLMOD", "GLMOD *", 0, 0, (void*)0, 0};
static swig_type_info _swigt__p_MARKOVLV = {"_p_MARKOVLV", "MARKOVLV *", 0, 0, (void*)0,
static swig_type_info _swigt__p_TABLESERVER = {"_p_TABLESERVER", "TABLESERVER *", 0, 0,
static swig_type_info _swigt__p_VAMOD = {"_p_VAMOD", "VAMOD *", 0, 0, (void*)0, 0};
static swig_type_info _swigt__p_VAPAR = {"_p_VAPAR", "VAPAR *", 0, 0, (void*)0, 0};
static swig_type_info _swigt__p_WIDDOWLV = {"_p_WIDDOWLV", "WIDDOWLV *", 0, 0, (void*)0,
static swig_type_info _swigt__p_char = {"_p_char", "char *", 0, 0, (void*)0, 0};

static swig_type_info *swig_type_initial[] = {
  &_swigt__p_ANNMOD,
  &_swigt__p_ANNUITYLV,
```

```
  &_swigt__p_ANNUITYLV2,
  &_swigt__p_CAPITALLV,
  &_swigt__p_FILE,
  &_swigt__p_GLMOD,
  &_swigt__p_MARKOVLV,
  &_swigt__p_TABLESERVER,
  &_swigt__p_VAMOD,
  &_swigt__p_VAPAR,
  &_swigt__p_WIDDOWLV,
  &_swigt__p_char,
};

static swig_cast_info _swigc__p_ANNMOD[] = {  {&_swigt__p_ANNMOD, 0, 0, 0},{0, 0, 0, 0}}
static swig_cast_info _swigc__p_ANNUITYLV[] = {  {&_swigt__p_ANNUITYLV, 0, 0, 0},{0, 0,
static swig_cast_info _swigc__p_ANNUITYLV2[] = {  {&_swigt__p_ANNUITYLV2, 0, 0, 0},{0, 0
static swig_cast_info _swigc__p_CAPITALLV[] = {  {&_swigt__p_CAPITALLV, 0, 0, 0},{0, 0,
static swig_cast_info _swigc__p_FILE[] = {  {&_swigt__p_FILE, 0, 0, 0},{0, 0, 0, 0}};
static swig_cast_info _swigc__p_GLMOD[] = {  {&_swigt__p_GLMOD, 0, 0, 0},{0, 0, 0, 0}};
static swig_cast_info _swigc__p_MARKOVLV[] = {  {&_swigt__p_WIDDOWLV, _p_WIDDOWLVTo_p_MA
{&_swigt__p_ANNMOD, _p_ANNMODTo_p_MARKOVLV, 0, 0},  {&_swigt__p_GLMOD, _p_GLMODTo_p_MARK
{&_swigt__p_VAMOD, _p_VAMODTo_p_MARKOVLV, 0, 0},  {&_swigt__p_CAPITALLV, _p_CAPITALLVTo_
{&_swigt__p_ANNUITYLV2, _p_ANNUITYLV2To_p_MARKOVLV, 0, 0},  {&_swigt__p_MARKOVLV, 0, 0,
{&_swigt__p_ANNUITYLV, _p_ANNUITYLVTo_p_MARKOVLV, 0, 0},{0, 0, 0, 0}};
static swig_cast_info _swigc__p_TABLESERVER[] = {  {&_swigt__p_TABLESERVER, 0, 0, 0},{0,
static swig_cast_info _swigc__p_VAMOD[] = {  {&_swigt__p_VAMOD, 0, 0, 0},{0, 0, 0, 0}};
static swig_cast_info _swigc__p_VAPAR[] = {  {&_swigt__p_VAPAR, 0, 0, 0},{0, 0, 0, 0}};
static swig_cast_info _swigc__p_WIDDOWLV[] = {  {&_swigt__p_WIDDOWLV, 0, 0, 0},{0, 0, 0,
static swig_cast_info _swigc__p_char[] = {  {&_swigt__p_char, 0, 0, 0},{0, 0, 0, 0}};

static swig_cast_info *swig_cast_initial[] = {
  _swigc__p_ANNMOD,
  _swigc__p_ANNUITYLV,
  _swigc__p_ANNUITYLV2,
  _swigc__p_CAPITALLV,
  _swigc__p_FILE,
  _swigc__p_GLMOD,
  _swigc__p_MARKOVLV,
  _swigc__p_TABLESERVER,
  _swigc__p_VAMOD,
  _swigc__p_VAPAR,
  _swigc__p_WIDDOWLV,
  _swigc__p_char,
};


/* -------- TYPE CONVERSION AND EQUIVALENCE RULES (END) -------- */

static swig_const_info swig_const_table[] = {
{0, 0, 0, 0.0, 0, 0}};

#ifdef __cplusplus
}
#endif
/* -----------------------------------------------------------------------------
```

```
 * Type initialization:
 * This problem is tough by the requirement that no dynamic
 * memory is used. Also, since swig_type_info structures store pointers to
 * swig_cast_info structures and swig_cast_info structures store pointers back
 * to swig_type_info structures, we need some lookup code at initialization.
 * The idea is that swig generates all the structures that are needed.
 * The runtime then collects these partially filled structures.
 * The SWIG_InitializeModule function takes these initial arrays out of
 * swig_module, and does all the lookup, filling in the swig_module.types
 * array with the correct data and linking the correct swig_cast_info
 * structures together.
 *
 * The generated swig_type_info structures are assigned staticly to an initial
 * array. We just loop through that array, and handle each type individually.
 * First we lookup if this type has been already loaded, and if so, use the
 * loaded structure instead of the generated one. Then we have to fill in the
 * cast linked list. The cast data is initially stored in something like a
 * two-dimensional array. Each row corresponds to a type (there are the same
 * number of rows as there are in the swig_type_initial array). Each entry in
 * a column is one of the swig_cast_info structures for that type.
 * The cast_initial array is actually an array of arrays, because each row has
 * a variable number of columns. So to actually build the cast linked list,
 * we find the array of casts associated with the type, and loop through it
 * adding the casts to the list. The one last trick we need to do is making
 * sure the type pointer in the swig_cast_info struct is correct.
 *
 * First off, we lookup the cast->type name to see if it is already loaded.
 * There are three cases to handle:
 *  1) If the cast->type has already been loaded AND the type we are adding
 *     casting info to has not been loaded (it is in this module), THEN we
 *     replace the cast->type pointer with the type pointer that has already
 *     been loaded.
 *  2) If BOTH types (the one we are adding casting info to, and the
 *     cast->type) are loaded, THEN the cast info has already been loaded by
 *     the previous module so we just ignore it.
 *  3) Finally, if cast->type has not already been loaded, then we add that
 *     swig_cast_info to the linked list (because the cast->type) pointer will
 *     be correct.
 * ----------------------------------------------------------------------------- */

#ifdef __cplusplus
extern "C" {
#if 0
} /* c-mode */
#endif
#endif

#if 0
#define SWIGRUNTIME_DEBUG
#endif


SWIGRUNTIME void
SWIG_InitializeModule(void *clientdata) {
```

```
  size_t i;
  swig_module_info *module_head, *iter;
  int found;

  clientdata = clientdata;

  /* check to see if the circular list has been setup, if not, set it up */
  if (swig_module.next==0) {
    /* Initialize the swig_module */
    swig_module.type_initial = swig_type_initial;
    swig_module.cast_initial = swig_cast_initial;
    swig_module.next = &swig_module;
  }

  /* Try and load any already created modules */
  module_head = SWIG_GetModule(clientdata);
  if (!module_head) {
    /* This is the first module loaded for this interpreter */
    /* so set the swig module into the interpreter */
    SWIG_SetModule(clientdata, &swig_module);
    module_head = &swig_module;
  } else {
    /* the interpreter has loaded a SWIG module, but has it loaded this one? */
    found=0;
    iter=module_head;
    do {
      if (iter==&swig_module) {
        found=1;
        break;
      }
      iter=iter->next;
    } while (iter!= module_head);

    /* if the is found in the list, then all is done and we may leave */
    if (found) return;
    /* otherwise we must add out module into the list */
    swig_module.next = module_head->next;
    module_head->next = &swig_module;
  }

  /* Now work on filling in swig_module.types */
#ifdef SWIGRUNTIME_DEBUG
  printf("SWIG_InitializeModule: size %d\n", swig_module.size);
#endif
  for (i = 0; i < swig_module.size; ++i) {
    swig_type_info *type = 0;
    swig_type_info *ret;
    swig_cast_info *cast;

#ifdef SWIGRUNTIME_DEBUG
    printf("SWIG_InitializeModule: type %d %s\n", i, swig_module.type_initial[i]->name);
#endif

    /* if there is another module already loaded */
```

```
    if (swig_module.next != &swig_module) {
      type = SWIG_MangledTypeQueryModule(swig_module.next, &swig_module, swig_module.typ
    }
    if (type) {
      /* Overwrite clientdata field */
#ifdef SWIGRUNTIME_DEBUG
      printf("SWIG_InitializeModule: found type %s\n", type->name);
#endif
      if (swig_module.type_initial[i]->clientdata) {
        type->clientdata = swig_module.type_initial[i]->clientdata;
#ifdef SWIGRUNTIME_DEBUG
        printf("SWIG_InitializeModule: found and overwrite type %s \n", type->name);
#endif
      }
    } else {
      type = swig_module.type_initial[i];
    }

    /* Insert casting types */
    cast = swig_module.cast_initial[i];
    while (cast->type) {
      /* Don't need to add information already in the list */
      ret = 0;
#ifdef SWIGRUNTIME_DEBUG
      printf("SWIG_InitializeModule: look cast %s\n", cast->type->name);
#endif
      if (swig_module.next != &swig_module) {
        ret = SWIG_MangledTypeQueryModule(swig_module.next, &swig_module, cast->type->na
#ifdef SWIGRUNTIME_DEBUG
        if (ret) printf("SWIG_InitializeModule: found cast %s\n", ret->name);
#endif
      }
      if (ret) {
        if (type == swig_module.type_initial[i]) {
#ifdef SWIGRUNTIME_DEBUG
          printf("SWIG_InitializeModule: skip old type %s\n", ret->name);
#endif
          cast->type = ret;
          ret = 0;
        } else {
          /* Check for casting already in the list */
          swig_cast_info *ocast = SWIG_TypeCheck(ret->name, type);
#ifdef SWIGRUNTIME_DEBUG
          if (ocast) printf("SWIG_InitializeModule: skip old cast %s\n", ret->name);
#endif
          if (!ocast) ret = 0;
        }
      }

      if (!ret) {
#ifdef SWIGRUNTIME_DEBUG
        printf("SWIG_InitializeModule: adding cast %s\n", cast->type->name);
#endif
        if (type->cast) {
```

```
          type->cast->prev = cast;
          cast->next = type->cast;
        }
        type->cast = cast;
      }
      cast++;
    }
    /* Set entry in modules->types array equal to the type */
    swig_module.types[i] = type;
  }
  swig_module.types[i] = 0;

#ifdef SWIGRUNTIME_DEBUG
  printf("**** SWIG_InitializeModule: Cast List ******\n");
  for (i = 0; i < swig_module.size; ++i) {
    int j = 0;
    swig_cast_info *cast = swig_module.cast_initial[i];
    printf("SWIG_InitializeModule: type %d %s\n", i, swig_module.type_initial[i]->name);
    while (cast->type) {
      printf("SWIG_InitializeModule: cast type %s\n", cast->type->name);
      cast++;
      ++j;
    }
    printf("---- Total casts: %d\n",j);
  }
  printf("**** SWIG_InitializeModule: Cast List ******\n");
#endif
}

/* This function will propagate the clientdata field of type to
* any new swig_type_info structures that have been added into the list
* of equivalent types.  It is like calling
* SWIG_TypeClientData(type, clientdata) a second time.
*/
SWIGRUNTIME void
SWIG_PropagateClientData(void) {
  size_t i;
  swig_cast_info *equiv;
  static int init_run = 0;

  if (init_run) return;
  init_run = 1;

  for (i = 0; i < swig_module.size; i++) {
    if (swig_module.types[i]->clientdata) {
      equiv = swig_module.types[i]->cast;
      while (equiv) {
        if (!equiv->converter) {
          if (equiv->type && !equiv->type->clientdata)
          SWIG_TypeClientData(equiv->type, swig_module.types[i]->clientdata);
        }
        equiv = equiv->next;
      }
    }
```

```
  }
}

#ifdef __cplusplus
#if 0
{
  /* c-mode */
#endif
}
#endif



#ifdef __cplusplus
extern "C" {
#endif

  /* Python-specific SWIG API */
#define SWIG_newvarlink()                             SWIG_Python_newvarlink()
#define SWIG_addvarlink(p, name, get_attr, set_attr)  SWIG_Python_addvarlink(p, name, ge
#define SWIG_InstallConstants(d, constants)           SWIG_Python_InstallConstants(d, co

  /* -----------------------------------------------------------------------------
   * global variable support code.
   * ----------------------------------------------------------------------------- */

  typedef struct swig_globalvar {
    char       *name;                  /* Name of global variable */
    PyObject *(*get_attr)(void);       /* Return the current value */
    int       (*set_attr)(PyObject *); /* Set the value */
    struct swig_globalvar *next;
  } swig_globalvar;

  typedef struct swig_varlinkobject {
    PyObject_HEAD
    swig_globalvar *vars;
  } swig_varlinkobject;

  SWIGINTERN PyObject *
  swig_varlink_repr(swig_varlinkobject *SWIGUNUSEDPARM(v)) {
    return PyString_FromString("<Swig global variables>");
  }

  SWIGINTERN PyObject *
  swig_varlink_str(swig_varlinkobject *v) {
    PyObject *str = PyString_FromString("(");
    swig_globalvar  *var;
    for (var = v->vars; var; var=var->next) {
      PyString_ConcatAndDel(&str,PyString_FromString(var->name));
      if (var->next) PyString_ConcatAndDel(&str,PyString_FromString(", "));
    }
    PyString_ConcatAndDel(&str,PyString_FromString(")"));
    return str;
  }
```

```
SWIGINTERN int
swig_varlink_print(swig_varlinkobject *v, FILE *fp, int SWIGUNUSEDPARM(flags)) {
  PyObject *str = swig_varlink_str(v);
  fprintf(fp,"Swig global variables ");
  fprintf(fp,"%s\n", PyString_AsString(str));
  Py_DECREF(str);
  return 0;
}

SWIGINTERN void
swig_varlink_dealloc(swig_varlinkobject *v) {
  swig_globalvar *var = v->vars;
  while (var) {
    swig_globalvar *n = var->next;
    free(var->name);
    free(var);
    var = n;
  }
}

SWIGINTERN PyObject *
swig_varlink_getattr(swig_varlinkobject *v, char *n) {
  PyObject *res = NULL;
  swig_globalvar *var = v->vars;
  while (var) {
    if (strcmp(var->name,n) == 0) {
      res = (*var->get_attr)();
      break;
    }
    var = var->next;
  }
  if (res == NULL && !PyErr_Occurred()) {
    PyErr_SetString(PyExc_NameError,"Unknown C global variable");
  }
  return res;
}

SWIGINTERN int
swig_varlink_setattr(swig_varlinkobject *v, char *n, PyObject *p) {
  int res = 1;
  swig_globalvar *var = v->vars;
  while (var) {
    if (strcmp(var->name,n) == 0) {
      res = (*var->set_attr)(p);
      break;
    }
    var = var->next;
  }
  if (res == 1 && !PyErr_Occurred()) {
    PyErr_SetString(PyExc_NameError,"Unknown C global variable");
  }
  return res;
}
```

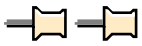
```
  SWIGINTERN PyTypeObject*
  swig_varlink_type(void) {
    static char varlink__doc__[] = "Swig var link object";
    static PyTypeObject varlink_type;
    static int type_init = 0;
    if (!type_init) {
      const PyTypeObject tmp
      = {
        PyObject_HEAD_INIT(NULL)
        0,                                  /* Number of items in variable part (ob_size
        (char *)"swigvarlink",              /* Type name (tp_name) */
        sizeof(swig_varlinkobject),         /* Basic size (tp_basicsize) */
        0,                                  /* Itemsize (tp_itemsize) */
        (destructor) swig_varlink_dealloc,   /* Deallocator (tp_dealloc) */
        (printfunc) swig_varlink_print,     /* Print (tp_print) */
        (getattrfunc) swig_varlink_getattr, /* get attr (tp_getattr) */
        (setattrfunc) swig_varlink_setattr, /* Set attr (tp_setattr) */
        0,                                  /* tp_compare */
        (reprfunc) swig_varlink_repr,       /* tp_repr */
        0,                                  /* tp_as_number */
        0,                                  /* tp_as_sequence */
        0,                                  /* tp_as_mapping */
        0,                                  /* tp_hash */
        0,                                  /* tp_call */
        (reprfunc)swig_varlink_str,        /* tp_str */
        0,                                  /* tp_getattro */
        0,                                  /* tp_setattro */
        0,                                  /* tp_as_buffer */
        0,                                  /* tp_flags */
        varlink__doc__,                     /* tp_doc */
        0,                                  /* tp_traverse */
        0,                                  /* tp_clear */
        0,                                  /* tp_richcompare */
        0,                                  /* tp_weaklistoffset */
#if PY_VERSION_HEX >= 0x02020000
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* tp_iter -> tp_weaklist */
#endif
#if PY_VERSION_HEX >= 0x02030000
        0,                                  /* tp_del */
#endif
#ifdef COUNT_ALLOCS
        0,0,0,0                             /* tp_alloc -> tp_next */
#endif
      };
      varlink_type = tmp;
      varlink_type.ob_type = &PyType_Type;
      type_init = 1;
    }
    return &varlink_type;
  }

  /* Create a variable linking object for use later */
  SWIGINTERN PyObject *
```

```
SWIG_Python_newvarlink(void) {
  swig_varlinkobject *result = PyObject_NEW(swig_varlinkobject, swig_varlink_type());
  if (result) {
    result->vars = 0;
  }
  return ((PyObject*) result);
}

SWIGINTERN void
SWIG_Python_addvarlink(PyObject *p, char *name, PyObject *(*get_attr)(void), int (*set
  swig_varlinkobject *v = (swig_varlinkobject *) p;
  swig_globalvar *gv = (swig_globalvar *) malloc(sizeof(swig_globalvar));
  if (gv) {
    size_t size = strlen(name)+1;
    gv->name = (char *)malloc(size);
    if (gv->name) {
      strncpy(gv->name,name,size);
      gv->get_attr = get_attr;
      gv->set_attr = set_attr;
      gv->next = v->vars;
    }
  }
  v->vars = gv;
}

SWIGINTERN PyObject *
SWIG_globals(void) {
  static PyObject *_SWIG_globals = 0;
  if (!_SWIG_globals) _SWIG_globals = SWIG_newvarlink();
  return _SWIG_globals;
}

/* -----------------------------------------------------------------------------
 * constants/methods manipulation
 * ----------------------------------------------------------------------------- */

/* Install Constants */
SWIGINTERN void
SWIG_Python_InstallConstants(PyObject *d, swig_const_info constants[]) {
  PyObject *obj = 0;
  size_t i;
  for (i = 0; constants[i].type; ++i) {
    switch(constants[i].type) {
    case SWIG_PY_POINTER:
      obj = SWIG_NewPointerObj(constants[i].pvalue, *(constants[i]).ptype,0);
      break;
    case SWIG_PY_BINARY:
      obj = SWIG_NewPackedObj(constants[i].pvalue, constants[i].lvalue, *(constants[i]
      break;
    default:
      obj = 0;
      break;
    }
    if (obj) {
```

```
        PyDict_SetItemString(d, constants[i].name, obj);
        Py_DECREF(obj);
      }
    }
  }

  /* -----------------------------------------------------------------------------*/
  /* Fix SwigMethods to carry the callback ptrs when needed */
  /* -----------------------------------------------------------------------------*/

  SWIGINTERN void
  SWIG_Python_FixMethods(PyMethodDef *methods,
    swig_const_info *const_table,
    swig_type_info **types,
    swig_type_info **types_initial) {
    size_t i;
    for (i = 0; methods[i].ml_name; ++i) {
      const char *c = methods[i].ml_doc;
      if (c && (c = strstr(c, "swig_ptr: "))) {
        int j;
        swig_const_info *ci = 0;
        const char *name = c + 10;
        for (j = 0; const_table[j].type; ++j) {
          if (strncmp(const_table[j].name, name,
              strlen(const_table[j].name)) == 0) {
            ci = &(const_table[j]);
            break;
          }
        }
        if (ci) {
          size_t shift = (ci->ptype) - types;
          swig_type_info *ty = types_initial[shift];
          size_t ldoc = (c - methods[i].ml_doc);
          size_t lptr = strlen(ty->name)+2*sizeof(void*)+2;
          char *ndoc = (char*)malloc(ldoc + lptr + 10);
          if (ndoc) {
            char *buff = ndoc;
            void *ptr = (ci->type == SWIG_PY_POINTER) ? ci->pvalue : 0;
            if (ptr) {
              strncpy(buff, methods[i].ml_doc, ldoc);
              buff += ldoc;
              strncpy(buff, "swig_ptr: ", 10);
              buff += 10;
              SWIG_PackVoidPtr(buff, ptr, ty->name, lptr);
              methods[i].ml_doc = ndoc;
            }
          }
        }
      }
    }
  }

#ifdef __cplusplus
}
```

```
#endif

/* -----------------------------------------------------------------------------*
 *  Partial Init method
 * -----------------------------------------------------------------------------*/

#ifdef __cplusplus
extern "C"
#endif
SWIGEXPORT void SWIG_init(void) {
  PyObject *m, *d;

  /* Fix SwigMethods to carry the callback ptrs when needed */
  SWIG_Python_FixMethods(SwigMethods, swig_const_table, swig_types, swig_type_initial);

  m = Py_InitModule((char *) SWIG_name, SwigMethods);
  d = PyModule_GetDict(m);

  SWIG_InitializeModule(0);
  SWIG_InstallConstants(d,swig_const_table);


}
```

Chapter 17
Examples



```
import markovlv
print "ALTERSRENTE"
a = markovlv.ANNUITYLV()
a.vSetStopTime(20)
a.vSetStartTime(120)
a.vSetSAge(65)
a.dSetPre(0.625)
temp = a.dSetBaseYear(2010)
temp = a.dSetActualYear(2010)
t0 = 1964
t1 = 2010
for i in range(120):
    a.dSetDisc(t0+i, 0.97)
    a.dSetQx(i,0.0075)

print "%s -- %-10s  %-10s" % ("age","DK","CF")
for i in range(50):
    print "%d -- %10.2f  %10.4f" % (i+t1-t0, a.dGetDK(i+t1-t0), a.dGetCF(i+t1-t0))

import markovlv
print "ALTERSRENTE2"
a = markovlv.ANNUITYLV2()
a.vSetStopTime(20)
a.vSetStartTime(120)
a.vSetSAge1(65)
a.vSetSAge2(65)
a.dSetPre(0.625)
temp = a.dSetBaseYear(2010)
temp = a.dSetActualYear(2010)
t0 = 1964
t1 = 2010
for i in range(120):
    a.dSetDisc(t0+i, 0.97)
    a.dSetQx1(i,0.0075)
    a.dSetQx2(i,0.0075)
a.dSetBenefit(0, 1)
a.dSetBenefit(1, 1)
a.dSetBenefit(2, 1)
```

```
print "%s -- %-30s  %-10s" % ("age","DK1--2","CF")
for i in range(50):
    print "%d -- %10.2f - %10.2f - %10.2f  %10.4f" % (i+t1-t0, a.dGetDK(i+t1-t0,0), a.dG

print "CAPITAL"
a = markovlv.CAPITALLV()
a.vSetStopTime(20)
a.vSetStartTime(70)
a.vSetSurvival(65,100000.)
a.vSetDeath(200000.)
a.vSetPremium(2000.)
temp = a.dSetBaseYear(2010)
temp = a.dSetActualYear(2010)
t0 = 1964
t1 = 2010
for i in range(120):
    a.dSetDisc(t0+i, 0.97)
    a.dSetQx(i,0.0075)

print "%s -- %-10s  %-10s" % ("age","DK","CF")
for i in range(50):
    print "%d -- %10.2f  %10.4f" % (i+t1-t0, a.dGetDK(i+t1-t0), a.dGetCF(i+t1-t0))

print "WIDDOW"
a = markovlv.WIDDOWLV()
a.vSetStopTime(20)
a.vSetStartTime(110)
a.dSetPre(0.625)
temp = a.dSetBaseYear(2010)
temp = a.dSetActualYear(2010)
t0 = 1964
t1 = 2010
for i in range(120):
    a.dSetDisc(t0+i, 0.97)
    a.dSetQx(i,0.0075)
    a.dSetQy(i,0.0050)
    a.dSetHx(i,0.75)
    a.dSetYx(i,i-3.)

print "%s -- %-10s  %-10s" % ("age","DK","CF")
for i in range(50):
    print "%d -- %10.2f  %10.4f" % (i+t1-t0, a.dGetDK(i+t1-t0), a.dGetCF(i+t1-t0))

a=markovlv.TABLESERVER()
a.vSetTable("CH-QX-EKM-1995")
print "Table Nr %d" % (a.iTableNumber())
print "x0 %d  xOmega %d  gender %d" % (a.iX0(),a.iXOmega(),a.iGender())
print " Tech I %10.6f" %(a.dITech())
for i in range(100):
   print "%d -- %10.4f" % (i, a.dGetValue(i))
a.vSetTable("RA-QX-RA-1857")
print "Table Nr %d" % (a.iTableNumber())
print "x0 %d  xOmega %d  gender %d" % (a.iX0(),a.iXOmega(),a.iGender())
```

```
print " Tech I %10.6f" %(a.dITech())
for i in range(110):
   print "%d -- %10.4f" % (i, a.dGetValue(i))
a.vSetTable("GB-QX-MEDIEVAL")
print "Table Nr %d" % (a.iTableNumber())
print "x0 %d  xOmega %d  gender %d" % (a.iX0(),a.iXOmega(),a.iGender())
print " Tech I %10.6f" %(a.dITech())
for i in range(100):
   print "%d -- %10.4f" % (i, a.dGetValue(i))

print a.pcAllTarifs()

##############################################################################################
#
#
##############################################################################################


import os
import sys
import math
import string
import math
import time
import markovlv
from scipy import *
import scipy.io
import scipy.stats
import numpy
import pylab as p

class Model2:

    def __init__(self):
        self.Now = time.clock()
        print 'Calculation of an annuity portfolio: Every Year differently'
# ================================================
        self.v = 1 / 1.035   # corresponding to 10 yr euro bond rate
        self.t = 2006
        self.sigma = 0.07
        print 'Sigma:' , self.sigma
        self.sigma = input(' New Sigma ? ')
        self.mu = -self.sigma*self.sigma/2
        self.NrSim = 15
        self.NrSim = input(' Nr of Simulations ? ')
        self.MaxT  = 50
        print 'Mu =',self.mu,' Sigma =', self.sigma, 'Nr Sim:', self.NrSim
        self.AnnuityName='annuity.dat'
        self.OutFile = 'out.txt'
        self.HowMany = 10
        self.ZCB = array([0.891975149642233 ,0.857779447242907 ,0.825175074968949 ,0.793

        print 'Useing Euro ZCB'
        self.Portfolio = scipy.io.read_array(self.AnnuityName)
        qx = scipy.io.read_array('qx_ER2000_2O.dat')
```

```
        qy = scipy.io.read_array('qy_ER2000_2O.dat')
        fx = scipy.io.read_array('fx_ER2000_2O.dat')
        fy = scipy.io.read_array('fy_ER2000_2O.dat')
# ================================================
        self.hm = markovlv.ANNUITYLV()
        self.hf = markovlv.ANNUITYLV()
        for j in range(qx.shape[0]):
            dTemp = self.hm.dSetQx(long(0.005+qx[j,0]), qx[j,1])
        for j in range(fx.shape[0]):
            dTemp = self.hm.dSetFx(long(0.005+fx[j,0]), fx[j,1])
        for j in range(2500):
            dTemp = self.hm.dSetDisc(j, self.v)

        self.hm.vSetStartTime(140)
        self.hm.dSetBaseYear(1993)
        self.hm.dSetActualYear(self.t)
        self.hm.dSetPre(0.625)
        self.hm.vSetG(0)

        for j in range(qy.shape[0]):
            dTemp = self.hf.dSetQx(long(0.005+qy[j,0]), qy[j,1])
        for j in range(fy.shape[0]):
            dTemp = self.hf.dSetFx(long(0.005+fy[j,0]), fy[j,1])
        for j in range(2500):
            dTemp = self.hf.dSetDisc(j, self.v)

        self.hf.vSetStartTime(140)
        self.hf.dSetBaseYear(1993)
        self.hf.dSetActualYear(self.t)
        self.hf.dSetPre(0.625)
        self.hf.vSetG(0)

        self.DK_stat = numpy.zeros((self.MaxT,1), dtype='f');
        self.CF_stat = numpy.zeros((self.MaxT,1), dtype='f');
        self.CF_Sim = numpy.zeros((self.MaxT,self.NrSim), dtype='f');
        self.MV_DK = numpy.zeros((1, self.NrSim), dtype='f');
        self.Dur = []

    def CalcExp(self):
        print 'Task 1 calculate statutory CF and DK'
        u = 0
        for i in range(self.Portfolio.shape[0]):
# Init Person i
            sex = self.Portfolio[i,2]
            if sex == 1:
                h = self.hm
            else:
                h = self.hf
            age = long(0.005+self.Portfolio[i,0])
            r   = self.Portfolio[i,1]
#            print sex, age, r, self.Portfolio[i,3]
            h.vSetStopTime(age)
            h.vSetSAge(long(0.005+self.Portfolio[i,3]))
            u = u + r * h.dGetDK(age)
```

```
# Get DK and CF Person i
           for j in range(self.MaxT):
               self.CF_stat[j, 0] = self.CF_stat[j, 0] + r * h.dGetCF(age + j);
        print u
        print 'time:', time.clock()-self.Now

    def CalcSim(self):
        print 'Task 2 Simulation of CF and DK'
        jetzt = time.clock()
        dNenner = 0.
        dZaehler = 0.
        for k in range(self.NrSim):
            XOmega = exp(self.mu * numpy.ones((100,1), dtype='f') + self.sigma * randn(1

            kktemp = double(1.0)

            for j in range (100):
                kktemp *= XOmega[j]
                dTemp = self.hm.dSetRelativeQxForTime(long(j), double(kktemp))
                dTemp = self.hf.dSetRelativeQxForTime(long(j), double(kktemp))

            for i in range(self.Portfolio.shape[0]):
                    sex = self.Portfolio[i,2]
                    if sex == 1:
                        h = self.hm
                    else:
                        h = self.hf
                    age = self.Portfolio[i,0]
                    r   = self.Portfolio[i,1]
#           print sex, age, r, self.Portfolio[i,3]
                    h.vSetStopTime(long(0.005+age))
                    h.vSetSAge(long(0.005+self.Portfolio[i,3]))
#                    print i
                    for j in range(self.MaxT):
                        self.CF_Sim[j,k] = self.CF_Sim[j,k] + r * h.dGetCF(long(0.005+ag
                        dNenner +=  r * h.dGetCF(long(0.005+age) + j) * self.ZCB[j]
                        dZaehler += r * h.dGetCF(long(0.005+age) + j) * self.ZCB[j] * j
            self.Dur.append(dZaehler / dNenner)
            if (mod(k,10) == 1) :
                abgelaufeneZeit = time.clock() - jetzt
                verbleibendeZeit = abgelaufeneZeit * (self.NrSim - k) / k
                print 'Simulation', k, 'out of ', self.NrSim,' Time elapsed ', time.cloc
            if (mod(k,self.HowMany) == 1):
                p.plot(range(self.MaxT),self.CF_Sim[range(self.MaxT),k])



################################################################################
def main():
################################################################################
        s = Model2()
        s.CalcExp()
        s.CalcSim()
```

```
        outFile = file(s.OutFile, 'w')
        scipy.io.write_array(outFile,  s.CF_Sim.transpose())
        outFile.close()
        print 'Done, time:', time.clock()-s.Now,'Sec.'
        p.figure(2)
        s.Dur.sort()
        xx=[]
        for i in range(len(s.Dur)):
            xx.append(i/(1.*len(s.Dur)))
#         print xx, s.Dur
        p.plot(xx,s.Dur)
        p.grid(True)
        p.show()
#    print 'Total lines read: ', self.totallines

if __name__ == "__main__":
    main()
```