RAPPORT DE STAGE MISSION
DEUXIÈME SEMESTRE

Développement de l'outil Hermès pour la tarification Groupe

Stagiaire :
Duc Hien VU

Encadrant :
Dimitri MINASSIAN

8 mai 2015

Remerciements

J'adresse mes sincères remerciements à l'équipe Actuariat Réassurance d'AXA Global P&C.

Je remercie plus particulièrement Dimitri MINASSIAN de m'avoir guidé dans mon travail et m'avoir aidé à trouver des solutions pour avancer. Sa disponibilité, son soutien, son encouragement et sa patience m'ont été précieux afin de bien mener mon travail.

Table des matières

Introduction

Ce stage est complémentaire au stage que j'ai effectué au premier semestre dans l'équipe Actuariat Réassurance, au sein d'AXA Global P&C. Durant ce deuxième stage, j'ai continué à travailler avec l'outil Hermès qui a pour vocation de modéliser les pertes atypiques et les pertes CAT et de tarifer les structures de réassurance. Cet outil est un grand programme R utilisant comme plateforme un logiciel (une surcouche R) développé en interne chez AXA. Il utilise la méthode de Monte Carlo pour simuler un grand nombre de scénarios de sinistralité.

Les missions accomplies pendant le premier stage sont les suivantes :
- S'approprier l'expertise AXA Global P&C en terme de tarification,
- Comprendre l'outil de tarification actuel et l'améliorer : identifier les parties de codes R pouvant être optimisée par le langage C/C++ et implémenter ces alternatives.
- Établir une hiérarchie des différents modules de l'outil existants et les rendre indépendants afin de permettre l'appel de ces derniers via la plateforme web d'interface utilisateur.

Les missions que j'ai réalisé pour le deuxième stage sont :
- Continuer à optimiser les codes Hermès sous R et C++,
- Développer de nouvelles fonctionnalités d'Hermès,
- Développer [1] de nouvelles méthodes de tarification (méthode historique, courbes Marché),
- Mesurer les impacts d'un changement de structure de réassurance,
- Vérifier la cohérence des résultats du nouvel outil et s'assurer de sa facilité d'utilisation par les utilisateurs.

Ce rapport est une brève présentation des missions mentionnées ci-dessus ainsi que les résultats obtenus. Je vais commencer par rappeler très rapidement les modifications que j'ai apportées à Hermès lors du premier stage en utilisant C/C++ et présenter les autres changements au niveau de code au cours du deuxième stage pour terminer la phase d'optimisation. Les résultats de tests entre deux versions d'Hermès (R vs R/C++) seront présentés pour évaluer l'efficacité de la nouvelle version en termes de temps de calcul. Je vais ensuite présenter les nouvelles fonctionnalités que j'ai développées pour la nouvelle version. La partie suivante mentionnera le backtesting et les difficultés rencontrées lors de la phase de test. Enfin, je vais parler des perspectives qu'on pourrait envisager sur Hermès.

1. cette mission est prise en charge par les autres membres de l'équipe, je n'ai défini que les inputs/outputs des fonctions

1 Contexte

La modélisation des risques joue un rôle très importante pour une entreprise en générale et plus particulièrement pour une compagnie d'assurance à cause de l'inversion du cycle de production qui entraîne une grosse incertitude sur les risques encourus. Elle aide les assureurs à bien tarifer les contrats d'assurance et de réassurance (et aussi de définir le montant de réserves, le niveau de capital requis, etc.). AXA Global P&C, au sein du groupe AXA, s'assure la tarification des traités de réassurance P&C afin de maitriser ses cessions aux acteurs externes de la réassurance. Elle permet d'optimiser les placements en réassurance et ainsi de protéger le groupe AXA selon son propre appétit au risque.

Dans ce contexte, un outil de modélisation des risques, que ce soit atypiques ou CAT et de tarification des structures de réassurance s'avère indispensable pour AXA. Un outil qui s'appelle Hermès a été développé par l'équipe Actuariat Réassurance d'AXA Global P&C pour répondre au besoin du groupe. Pourtant cet outil présente quelques limites :

- l'outil fonctionne sous une plateforme qui n'est pas utilisée par toutes les entités,
- le temps de calcul de l'outil est potentiellement long pour certains scénarios de modélisation,
- les résultats ne sont pas suffisamment stables avec le nombre de simulations Monte Carlo choisi, ce qui nous a forcé à augmenter le nombre de simulations et donc augmenter encore le temps de calcul.

Pour répondre à toutes ces limites, l'équipe a décidé de re-développer l'outil d'une manière plus puissante et de construire une plateforme web pour que l'outil puisse être utilisé facilement non seulement pour nous mais aussi pour toutes entités au niveau mondial. L'introduction de la plateforme web modifierait considérablement la façon dont Hermès fonctionne et nécessite donc une restructuration des codes et des inputs/outputs. L'optimisation du temps de calcul se ferait à deux niveaux. le premier niveau profite les avantages des progrès informatiques avec le calcul parallèle. Le deuxième niveau, qui intervient même avant le premier, est l'optimisation du code R de l'outil. Ma mission est donc de restructurer l'outil pour préparer son intégration dans la plateforme web et d'optimiser le temps de calcul d'Hermès au deuxième niveau. Enfin, j'ai aussi développé quelques nouvelles fonctionnalités en profitant les fonctions existantes dans l'outil.

2 Optimisation de l'outil Hermès

La version ancienne d'Hermès, codée purement sous R, présente des limites en temps de calcul avec les structures de matrices. Même avec les fonctions R du type lapply(), sapply(), etc. les calculs peuvent être longs si la dimension de la matrice considérée est grande. Comme mentionné dans le premier rapport, la réduction du temps de calcul est le plus grand défi pour Hermès, d'une part parce que la modélisation de sinistralité des branches longues (responsabilité civile par exemple) avec la prise en compte des cadencements de paiement et l'application des traités de réassurances sont longues, d'autre part parce que le nombre de simulations Monte Carlo devrait passer de 10000 à 50000 pour s'assurer la convergence des estimateurs. La solution qui nous parait la plus logique est le recours vers C++ en utilisant le package Rcpp et Rtools pour le compilateur C++.

Lors du premier stage, j'ai créé deux fonctions C++ [2] :

- AAD_AAL_Appli(NumericMatrix M, double AAD, double AAL) : pour l'application de AAD et AAL à la matrice de sinistres projetés dans le temps. Cette fonction est appelée dans la fonction R Layer_Cost().
- exhaustion_prob_CPP(NumericMatrix M, double AAD, double AAL, double N) : pour calculer la probabilité de consommer entièrement la portée (exhausted probability). Cette fonction est appelée dans la fonction R Layer_price2().

Pendant mon deuxième stage, j'ai continué à identifier les phases de codes R gourmandes en temps de calcul et les optimiser sous C++. Au total j'ai créé 11 petites fonctions C++ qui interviennent à plusieurs endroits du cœur R de Hermès [3].

Rappelons-nous que l'ordre de fonctionnement de Hermès est grosso modo : la modélisation des pertes (Atypiques ou CAT), l'application de la structure de réassurance (XS pour l'instant), le calcul des OEP/AEP [4] et le reporting. Je vais maintenant présenter les principes de fonctionnement de chacun des modules d'Hermès, ainsi que les changements au niveau du code pour l'optimiser.

2.1 Modélisation des pertes atypiques

2.1.1 Principes

Les points clés de la modélisation atypiques ont été présentés dans le premier rapport. J'expliquerai ci-dessous les parties de code Hermès correspondantes ainsi que les alternatives C++ que j'ai introduites :

- Liste des coûts et des fréquences
- Matrice des montants ultimes de sinistres
- Application des taux de change
- Application des cadencements de paiement (alternative C++)
  - Cadencements de paiement déterministes
  - Cadencements de paiement stochastiques (désactivés)
- Préparation pour la clause de stabilité (alternative C++)

2.1.2 Optimisation du code Hermès

Optimisation de la partie de code R qui génère la matrice de cadencements de paiement déterministes :

Rappelons-nous que, pour les branches d'activités "longues" telles que la responsabilité civile, la gestion des sinistres peut s'étaler sur une durée de plusieurs années (maximum 30 années dans Hermès). La charge ultime, après être modélisée, doit être réparties dans le temps suivant une décomposition de temps moyen de paiement pour un pourcentage cible de la charge ultime (fournie par le générateur de pertes). Deux méthodes d'estimation de la part de la charge ultime payée chaque année ont été implé-

2. les détails de ces deux fonctions peuvent être trouvés dans le premier rapport
3. Voir Annexe A1 pour le récapitulatif des fonctions C++ intervenant à chaque module Hermès
4. cf. partie 2.2.1 pour les notions OEP/AEP

mentées dans la version ancienne d'Hermès : le cadencement de paiement déterministe et le cadencement de paiement stochastique. Dans cette nouvelle version, nous ne gardons que la méthode déterministe, la méthode stochastique sera remplacée par une lecture directe d'une cadence fournie par le fichier générateur. J'ai donc désactivé la partie de code correspondante au cadencement de paiement stochastique et optimisé la fonction qui génère la matrice de cadencement de paiement déterministe.

Je rappelle ici le raisonnement du cadencement déterministe dans Hermès que j'ai déjà détaillé dans mon premier rapport. Un générateur founit un cadencement de paiements moyen en associant à des pourcentages cibles cumulés $(C_i)_{i=1,...,k}$ un temps moyen (encore appelé temps caractéristique) $(t_i)_{i=1,...,k}$. Nous voulons effectuer la démarche inverse en associant à chaque année de développement un pourcentage de paiements.

Nous commençons par estimer les temps X_i mis pour accomplir les pourcentages cibles cumulées C_i. Hermès suppose que $X_i \sim \mathcal{E}(\frac{1}{t_i})$. Ci-dessous un exemple :

i	C_i	t_i	x_i	x_i cumulé
1	0,05	0,2	x_1	x_1
2	0,25	0,8	x_2	$x_1 + x_2$
3	0,5	13	x_3	$x_1 + ... + x_3$
4	0,75	2,5	x_4	$x_1 + ... + x_4$
5	0,95	2,2	x_5	$x_1 + ... + x_5$
6	1	1,3	x_6	$x_1 + ... + x_6$

TABLE 1 – Exemple de cadencement de paiement

Hermès impose la condition que le temps total de paiement $\sum_{i=1}^{k} x_i$ est majoré par $T_{max} = \sum_{i=1}^{k} t_i$ (dans l'exemple ci-dessus, $T_{max} = 20$). Tant que les réalisations $(x_i)_{i=1,...,k}$ ne satisfont pas cette condition, Hermès retire les lois exponentielles.

Ensuite, pour un temps t donné entre 0 et T_{max}, exprimé en nombre d'années, le pourcentage p_t de sinistre payé est obtenu par interpolation linéaire sur les x_i cumulés. Reprenons l'exemple ci-dessus et supposons que $x_1 + x_2 < 1 < x_1 + x_2 + x_3$. Le pourcentage p_1 du montant ultime de sinistre payé au bout d'un an sera compris entre $C_2 = 0{,}25$ et $C_3 = 0{,}5$. Par interpolation linéaire,

$$p_1 = 0,25 + \frac{1-(x_1+x_2)}{x_3} \times (0,5 - 0,25)$$

Comme nous utilisons la méthode de simulation Monte Carlo, pour un pourcentage cible C_t, Hermès n'effectue pas un seul tirage mais N tirages indépendants de loi exponentielle $(x_i^n)_{n=1,...,N}$. Ainsi les p_t mentionnés ci-dessus sont les moyennes des $(p_t^n)_{n=1,...,N}$.

Nous obtenons une matrice de cadencements de paiements donc le nombre de lignes est G (le nombre de générateur) et le nombre de colonnes est *Payment_Pattern_Length* qui est le nombre maximum d'années de développements des générateurs utilisés. Nous appliquons cette matrice de cadencements de paiements aux montants ultimes de sinistres pour obtenir la ***matrice de paiements par année de développement***, dont le nombre de ligne et N et le nombre de colonne est (*Payment_Pattern_Length* + 2). Les

deux premières colonnes représentent le numéro de générateur et le numéro de simulation (year). Cette matrice est ordonnée par year.

La fonction principale de cette partie est Payment_Pattern_Calculation.mean() qui renvoie la matrice de cadencements de paiement. J'ai recodé cette fonction en utilisant quatres fonctions C++ supplémentaires : percentage_payment_CPP(), invcumsum2_CPP(), cumulation_all_CPP() et sum_per_row_CPP(). Elles sont appelées dans l'ordre suivante :

```
# Temps_cumules_Matrix=t(apply(Exp_Matrix,1,cumsum))
Temps_cumules_Matrix = cumulation_all_CPP(Exp_Matrix)

# calcul des pourcentages payes au bout de chaque annee par interpolation
    lineaire
# C++ here
M= percentage_payment_CPP(Temps_cumules_Matrix, Ci, Tmax, nb_pas, Nexp)
#Pourcentage cumule paye au bout de chaque annee
Pour_Pay=invcumsum2_CPP(M)
```

La fonction sum_per_row_CPP() remplace apply(. . . ,1,sum). Elle intervient dans la partie où Hermès cumule les réalisations de lois exponentielles pour calculer le temps total mis pour régler un sinistre. La fonction cumulation_all_CPP() est simplement une alternative C++ de tapply(. . . , cumsum) pour cumuler les pourcentages de paiement d'un sinistres à chaque année de développement. invcumsum2_CPP() est une alternative C++ de la fonction R invcumsum2() qui prend en argument une matrice de paiements cumulés et renvoie la matrice de paiements décumulés correspandante.

La fonction percentage_payment_CPP(Temps_cumulés_Matrix, Ci, Tmax, nb_pas, Nexp) remplace sapply(1 :Tmax,simu_pourt,Temps_cumulés_Matrix) avec simu_pourt() définie ci-dessus. L'idée est d'utiliser une interpolation linéaire pour estimer le pourcentage de paiements par année de développement.

Préparation pour la clause de stabilité [5]

Pour préparer l'application de la clause de stabilité aux montants de portée et priorité dans le module Pricing, il faut avoir une matrice de ratios. A l'étape précédente on a la matrice de paiements de sinistres projetés dans le temps. On établit dans cette étape :

– la matrice de paiements par année de développement cumulés
– la matrice de paiements par année de développement désindéxés cumulés

Pour calculer ces deux matrices plus rapidement, j'ai créé la fonction C++ cumulation_CPP(M) pour remplacer tapply(M, 1, cumsum). L'idée est la même que cumulation_all_CPP() sauf qu'ici on doit garder les deux première colonne de la matrice M comme référence du numéro de loi et du numéro de simulation.

5. cf. partie 1.1.4 du premier rapport pour le raisonnement

2.2 La modélisation CAT

2.2.1 Principes

La modélisation CAT n'utilise pas les générateurs fréquences-coûts comme la modélisation atypique. Ils existent plusieurs types de fichiers générateurs CAT mais le plus usuel est Year_Loss_Table avec une colonne pour l'année de survenance de sinisre, une colonne pour l'événement ID, les deux autres colonnes pour le montant moyen de sinistre et son écart-type. Le point particulier de la modélisation CAT est la co-existence de plusieurs visions du risque CAT (et donc plusieurs fichiers de générateurs). Pour chacun des périls, nous pouvons avoir au maximum 3 différents modèles vendeurs (notés successivement M1, M2 et M3 [6]) et une vision interne du groupe AXA (notée M4). Ce dernier n'est rien d'autre qu'un "blend" des trois modèles vendeurs. Il s'avère donc nécessaire de détailler comment les pertes CAT sont générées sous Hermès.

Chaque entité d'AXA dispose certains périls disponibles à modéliser. Chaque péril contient au moins un générateur M4 et potentiellement des autres générateurs des modèles vendeurs (M1, M2 et M3). L'utilisateur peut choisir plusieurs périls à modéliser, les modèles utilisés pour chaque péril et le coefficient de blending correspondant.

Pour expliquer le blending des modèles vendeurs, je commence par rappeler les notions **OEP (Occurence Exceedance Probability)** et **AEP (Aggregate Exceedance Probability)**. L'OEP représente la probabilité d'avoir un événement pendant une période donnée (généralement un an) dont le montant de pertes est au moins égal à un montant donné. L'AEP représente la probabilité d'avoir un montant de pertes totales de l'année au moins égal à un montant donné.

Nous utilisons en pratiques des courbes OEP et AEP qui sont liées à la notion de **période de retour**. Une période de retour, exprimée en années, correspond au temps moyen à attendre avant de voir revenir une année présentant des pertes au moins équivalentes. C'est l'inverse de la probabilité d'occurence pendant l'année. La courbe OEP associe une période de retour au coût maximal d'un événement sur une année tandis que la courbe AEP associe un période de retour au coût total des événements sur une année. Ces courbes sont déterminées à partir d'une table O/AEP. Le tableau d'O/AEPs est obtenus sous Hermès en ordonnant les pertes simulées dans l'ordre croissante (ce qui correspond à une fonction de répartition empirique) et prenant les quantiles correspondants pour les OEPs et les pertes cumulées pour les AEPs.

Return Period	OEP	AEP
1 year out of 2	1	5
1 year out of 5	2	8
1 year out of 10	5	12
1 year out of 50	7	15
1 year out of 100	10	20
1 year out of 200	15	22
1 year out of 500	20	30

TABLE 2 – exemple tableau O/AEPs

6. les noms des modèles vendeurs ont été modifiés pour la raison de confidentialité

Dans l'exemple ci-dessus, on attend une perte maximale de 1 million d'euros et une perte agrégée de 5 millions d'euros tous les 2 ans (i.e. probabilité d'avoir une perte maximale de 1 million d'euros et une perte aggrégée de 5 millions d'euros pendant l'année est 50%,etc.

Supposons maintenant qu'une certaine entité veut modéliser à la fois trois périls : Eurowind, Earthquake et Hail. L'utilisateur ont rempli le tableau de blending comme suivant :

Péril	M1	M2	M3	M4	Projection Factor
Eurowind	33	33	33	0	1
Earthquake	20	80	0	0	1,2
Hail	0	0	0	100	1

TABLE 3 – exemple CAT : table de blending

Hermès commence toujours par générer les pertes selon le fichier de généreateurs M4 (et calculer les OEPs correspondants). Après il va corriger les pertes (grosso modo par homothétie) afin d'obternir les OEPs voulus. Les OEPs voulus (OEPs cibles) sont obtenus par une combinaison linéaire des OEPs issus de trois modèles vendeurs avec les pondérations définies dans le tableau de blending. Dans l'exemple ci-dessus, les OEPs cibles du péril Eurowind est une combinaison de 33% des OEPs de chacun des modèles vendeurs, les OEPs cibles du péril Earthquake est 20% OEPs du M1 + 80% OEPs du M2, alors que pour Hail, nous ne faisons pas de blending.

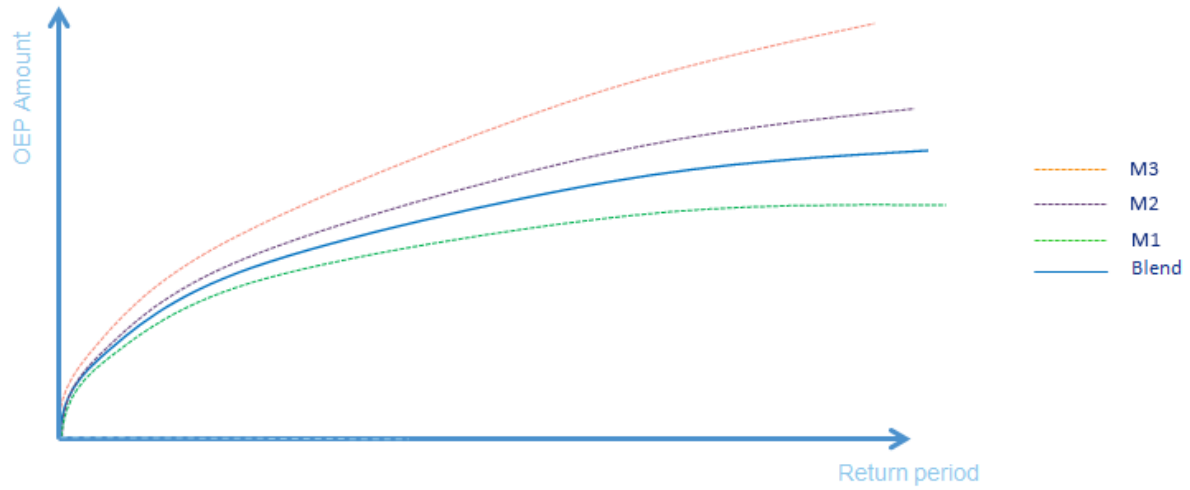

FIGURE 1 – exemple : OEPs' blending

Les "projection factors" sont des facteurs multiplicatifs aux montants de pertes pour représenter l'écart des visions de pertes entre le générateur et ce qui se réaliserait l'année prochaine. Un "projection factor"

égal à 1 indique qu'il n'y a aucune différence entre les deux visions.

2.2.2 Optimisation du code Hermès

La modélisation CAT ne prend pas en compte le cadencement de paiement, nous supposons que les sinistres sont réglés pendant l'année. Donc la fonction CAT_Modeling_func() est déjà assez rapide sous R. les alternatives C++ n'interviennent que dans la fonction EP_correction() où nous calculons les OEPs à partir des pertes simulées et ensuite corrigeons les pertes pour obtenir les OEPs cibles. Le calcul des maximums et des sommes cumulées sont plus rapide sous C++ que sous R.

Afin de rester cohérent avec la fonction NON_CAT_Modeling_func(), la fonction CAT_Modeling_func() comporte aussi une phase de préparation de la clause de stabilité en introduisant :

- la matrice de paiements par année de développement cumulés
- la matrice de paiements par année de développement désindéxés cumulés

L'ancienne version d'Hermès a repris le code de la fonction NON_CAT_Modeling_func() pour générer ces deux matrices mais cela est en fait inutile et ralentit la vitesse d'Hermès. Dans le cadre de la modélisation CAT (avec une seule année de paiement et donc pas de problème d'inflation), les deux matrices mentionnées ci-dessus sont les mêmes et identiques à la matrice de paiements non-cumulés. J'ai donc enlevé cette partie de code R et affecté la valeur de la matrice des paiements non-cumulés à ces deux matrices.

2.3 Application de traités XS

On applique la structure XS (limite, priorité, AAD, AAL) à travers les étapes suivantes :

- Calcul de la matrice de ratios (division terme à terme de la matrice de paiements par année de développement cumulés par la matrice de paiements par année de développement désindéxés cumulés),
- Charges du réassureur après application de portée et de priorité en tenant compte de la clause de stabilité (utilisation de C++),
- décumulation de la matrice des charges du réassureur. C'est une phase de préparation pour l'application d'AAD et AAL.

J'ai codé quelques fonctions C++ pour accélérer ces étapes :

• Fonction sum_per_year_CPP() : pour aggréger les paiements par numéro de simulation

• Fonction discount_rate_appli_CPP() : pour prendre en compte du taux d'actualisation.

Ces deux fonctions interviennent très souvent dans Hermès. Par exemple, dans la fonction «XS_Application_func() » :

```
1 Layer_Net_Cost_per_simulation=tapply(as.matrix(Layer_Net_Cost[,-(1:2)])%*%
    (1/Discount_Rate),Layer_Net_Cost[,2],sum)
```

sera remplacée par :

```
Layer_Net_Cost_per_simulation= as.vector(sum_per_year_CPP( discount_rate_
    appli_CPP(Layer_Net_Cost, Discount_Rate), 1, N))
```

Le script R ci-dessus applique un taux d'actualisation aux paiements par année de développement et ensuite agréger les paiements par numéro de simulation (variable year – 2è colonne de la matrice Layer_Net_Cost). Pour clarifier cette ligne de code, prenons-nous un exemple simple où le nombre d'années de développement est 3. Nous faisons 3 simulations (year = 1, 2 et 3).

Layer_Net_Cost :

$$\begin{pmatrix} (id) & (year) & (\text{Année N}) & (\text{Année N+1}) & (\text{Année N+2}) \\ 1 & 1 & 1 & 2 & 3 \\ 2 & 1 & 4 & 5 & 6 \\ 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 3 \\ 1 & 3 & 4 & 4 & 4 \\ 2 & 3 & 4 & 4 & 4 \end{pmatrix}$$

Nous prenons Discount_Rate = (1 1,2 1,5) le vecteur de taux d'actualisation correspondant aux trois années de développement. Supposons qu'il y a 2 sinistres lors de la première simulation, 3 sinistres lors de la deuxième simulation et 2 sinistres pour la troisième simulation. Au total nous avons simulé 7 sinistres. Nous appliquons le vecteur de taux d'actualisation aux paiements de chaque sinistre et fait la somme pour avoir le paiement agrégé, actualisé à l'année N :

Layer_Net_Cost :

$$\begin{pmatrix} (id) & (\text{monant agrégé}) \\ (1) & 4,667 \\ (2) & 12,167 \\ (3) & 5,000 \\ (4) & 5,000 \\ (5) & 5,667 \\ (6) & 10,000 \\ (7) & 10,000 \end{pmatrix}$$

Explication : Par exemple, pour le premier sinistre : 1(1/1) + 2*(1/1,2) + 3*(1 /1,5) = 4,667.*

Ensuite, on somme les paiements actualisés par numéro de simulation (year) en utilisant la fonction tapply() et en référençant à la colonne (year) de la matrice Layer_Net_Cost.

Layer_Net_Cost_per_simulation :

$$\begin{pmatrix} (\text{year}) & (\text{montant}) \\ (1) & 16,833 \\ (2) & 15,667 \\ (3) & 20,000 \end{pmatrix}$$

Cette méthode est optimale sous R à mon point de vue, on a utilisé la fonction tapply() pour éviter les boucles for. Néanmoins, ça prend du temps de calcul avec les vraies données générées dans Hermès (exemple : le nombre de simulation est 50000, le nombre d'années de développement est 30). L'alternative C++ avec deux fonctions discount_rate_appli_CPP() et sum_per_year_CPP() que j'ai présentées ci-dessus est beaucoup plus efficace en termes de temps de calcul.

• Fonction limit_priority_with_index_clause_appli_CPP().

Dans la fonction Layer_Cost() ,

```
Reinsurance_Cost_cumulated[,3:ncol(Reinsurance_Cost_cumulated)]<-pmax(pmin(
    Gross_Cost_Matrix[,3:ncol(Gross_Cost_Matrix)]-Ratio_Matrix*Layer_details
    $priority,Ratio_Matrix*Layer_details$limit),0)
```

est remplacée par

```
Reinsurance_Cost_cumulated = limit_priority_with_index_clause_appli_CPP(
    Gross_Cost_Matrix, Ratio_Matrix, Layer_details$limit, Layer_details$
    priority)
```

L'idée est d'appliquer la clause de stabilité à la portée et la priorité et ensuite appliquer les portées et les priorités corrigées par année de développement à la matrice de sinistres projetés dans le temps.

• Fonction percentage_payment_CPP() : détaillée au-dessus dans la partie 1.1.2.

J'ai également recodé quelques fonctions simples telles que apply(. . . , sum), apply(. . . , cumsum) et invcumsum() (pour décumuler une matrice de paiements) sous C++ pour gagner davantage du temps de calcul : cumulation_CPP(), sum_per_row_CPP(), invcumsum2_CPP(), division_CPP().

2.4 Calcul des OEPs et AEPs

Cette module calcule le tableau des O/AEPs bruts (avant réassurance), celui des O/AEPs nets (après réassurance) et aussi les O/AEPs de chacun des lois de générateurs (les lois de générateurs dans la modélisation atypique et les périls dans la modélisation CAT).

Le calcul des OEPs et AEPs bruts/nets est accéléré comme dans la fonction EP_Correction() en utilisant deux fonctions C++ : max_per_year_CPP() et sum_per_year_CPP()

L'optimisation du calcul du tableau O/AEPs par loi de générateurs s'est faite non seulement grâce aux alternatives C++ mais aussi par la façon dont on identifie les pertes de chaque générateurs sous R. L'ancienne version d'Hermès a pris la matrice des pertes (dont la première colonne est le numéro de simulation, la deuxième colonne est le numéro de lois et la troisième colonne est le montant de pertes), re-ordonné cette matrice suivant la deuxième colonne et scindé cette matrice en plusieurs sous-matrices (une sous-matrice par loi de générateurs). Ensuite Hermès calcule les O/AEPs pour chaque sous-matrice de pertes. Pour éviter le ré-ordonnancement et le "split" de la matrice des pertes, j'ai calculé les O/AEPs de chaque loi de générateurs à partir de la liste des coûts et des fréquences (list_of_costs_and_frequencies). Cette liste a été générée dans la phase de modélisation des pertes (atypiques ou CAT) et c'est à partir de cette liste que Hermès a construit la matrice des pertes.

2.5 Comparaison du temps de calcul entre deux versions Hermès : R et R/C++ :

J'ai effectué les tests sous 2 versions Hermès avec 3 lois de générateurs atypiques de branche longue (30 années de développement) et un traité XS à 2 layers (le nombre de simulations N est successivement 10000, 20000, 30000, 40000 et 50000).

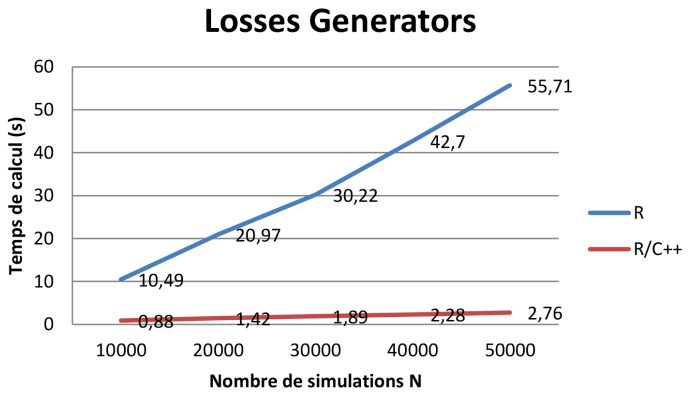

FIGURE 2 – Comparaison du temps pour générer les pertes

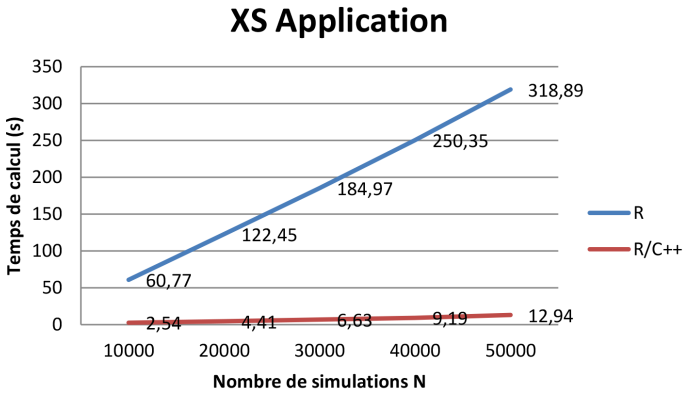

FIGURE 3 – Comparaison du temps pour le pricing XS

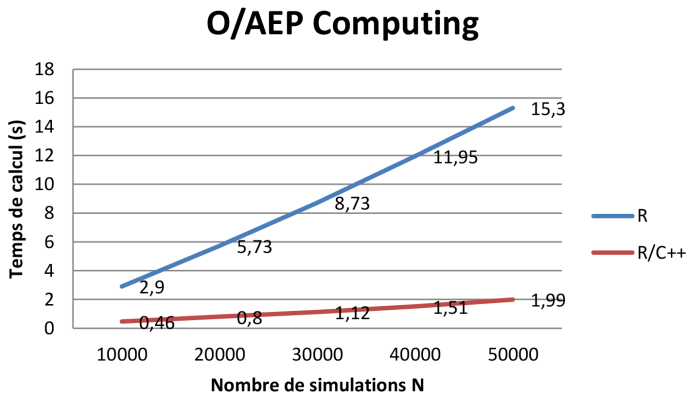

FIGURE 4 – Comparaison du temps pour calculer les O/AEPs

Nous pouvons voir que Hermès version R/C++ est beaucoup plus rapide pour chacun des modules principaux. L'économie du temps de calcul est d'ordre 95%.

N	10000	20000	30000	40000	50000
R	74,16	149,15	223,92	305	389,9
R/C++	3,88	6,63	9,64	12,98	17,69
Temps économisé	94,77%	95,55%	95,69%	95,74%	95,46%

TABLE 4 – Le temps de calcul total de Hermès (en s)

L'économie en temps de calcul pour la modélisation CAT est un peu moins forte que celle pour la modélisation atypique mais reste quand même très significative.

3 Développement de nouvelles fonctionnalités d'Hermès

Après avoir optimisé les modules déjà existants dans l'ancienne version d'Hermès, j'ai développé quelques nouvelles fonctionnalités. Nous ne gardons plus les structures QS et XS-on-QS-retention dans la deuxième version d'Hermès. Par contre, nous avons introduit une nouvelle structure XS-Aggregate pour se couvrir contre les pertes CAT à fréquences assez élevées. J'ai également créé un module pour faire les mises à jour des lois de générateurs et un module pour la contribution des modèles vendeurs dans la modélisation CAT.

3.1 XS Aggregate

La structure XS Aggregate a été introduite pour faire face à des sinistres CAT à fréquences élevées. Elle est utilisée en complémentaire de la structure XS. L'idée est simple : si on applique la structure XS

Aggregate à un ensemble de périls (défini par l'utilisateur), on cumulera les pertes de ces périls, ensuite on va retenir les pertes éligibles pour appliquer un traité XS avec un ensemble de paramètre (portée, priorité, nombre de reconstitutions, AAD, AAL) comme dans un programme XS.

On envisage plusieurs conditions d'éligibilité possibles des pertes agrégées suivant le type d'inputs définis par l'utilisateur :

- plafond p (cap) : le montant des pertes agrégées sont limité à p,
- seuil exclu s_e (Excluded Threshold) : les pertes agrégées dont le montant dépassent s_e ne sont pas prises en compte,
- franchise déduite f_d (deducted franchise) : les pertes agrégées d'un montant au-dessus de f_d sont réduites de f_d,
- franchise atteinte (reached franchise) f_a : les pertes agrégées d'un montant en-dessous de f_a ne sont pas prises en compte.

L'utilisateur peut fournir soit p soit s_e (l'autre est fixé à 0 par défaut) et soit f_d soit f_a (l'autre est fixé à Infini par défaut). Après avoir les pertes éligibles, on applique une structure XS classique. Il suffit d'appeler la fonction XS_Application_func() avec les bons paramètres au vecteur des pertes éligibles. Dans l'exemple ci-dessus, si la droite verte correspond au plafond p, la perte agrégée S1 sera capée à 9,

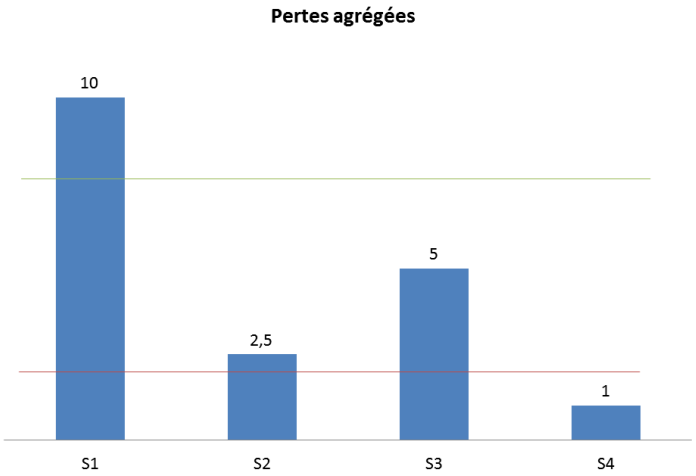

FIGURE 5 – exemple : XS Aggregate

tandis que si cette droite correspond au seuil exclu s_e, la perte agrégée S1 ne sera pas prise en compte pour la structure XS Aggregate. Si la droite rouge correspond à la franchise déduite, la perte S4 ne sera pas prise en compte, les trois autres pertes devra être réduites de 2 avant rentrer dans la structure XS Aggregate. Si la droite rouge correspond à la franchise atteinte, la perte S4 ne sera pas prise en compte, les trois autres pertes rentrent toutes dans la structure XS Aggregate avec leur montant "initiale" (en oubliant le cap/seuil exclu ici).

J'ai également calculé la part des pertes éligibles pour chacun des périls à appliquer la structure XS Aggregate pour voir si un péril en consomme beaucoup.

3.2 Mise à jour des lois de générateurs

La mise à jour ici désigne l'évolution des primes pures et primes techniques de l'année N-1 à l'année N, passant par les étapes intermédiaires pour passer du fichier génrateurs de l'année N-1 à celui de l'année N.

3.2.1 Cas atypique

Pour évaluer l'évolution des primes pures et primes techniques de l'année N-1 à l'année N, nous allons mettre à jour les lois de générateurs. Plusieurs scénarios sont envisageables.

- Si le fichier de générateurs de l'année N ne contient qu'une seule loi de générateurs et
  - Si le fichier de générateurs de l'année N-1 a aussi cette seule loi de génrateurs : Hermès fera la tarification N-1 et ensuite mettra à jour la loi de fréquence, la loi de sévérité et enfin les autres paramètres du fichier générateur pour arriver à la tarification N.
  - Si le fichier de générateurs de l'année N-1 ne contient pas la loi de générateur de l'année N, Hermès ne fera rien.
  - Si le fichier de générateurs de l'année N-1 contient une loi de générateurs de l'année N et d'autre(s) loi(s) de générateurs : Hermès fera la tarification N-1 avec toutes les lois disponibles pour l'année N-1, ensuite mettra à jour la loi présente en année N (mise à jour de la loi de fréquences et après la loi de sévérités), enlèvera les lois de générateurs non-présentes en N et enfin mettra à jour les autres paramètres pour arriver à la tarification N.
- Si le fichier de générateurs N contient plusieurs lois de génrateurs et
  - Si le fichier de générateurs N-1 contient au moins une loi en commun : Hermès fera la tarification N-1 et mettra à jour successivement les lois en commun et enfin retirer les lois présentes en N-1 mais non-présentes en N et rajouter les lois présentes en N mais non-présentes en N-1 pour arriver à la tarification N.
  - Si le fichier de générateurs N-1 contient aucune loi en commun : Hermès ne fera rien.

3.2.2 Cas CAT

Pour évaluer l'évolution des primes pures et primes techniques de l'année N-1 à l'année N, nous allons mettre à jour successivement les périls.

- Si l'ensemble des périls disponibles en N-1 en contient aucun péril modélisé en N : Hermès ne fera pas de mise à jour.
- Si l'ensemble des périls disponibles en N-1 contient au moins un péril modélisé en N : Hermès fera la tarification N-1 avec les périls en commun et ensuite mettra à jour successivement ces périls en commun, rajouter les périls modélisés en N mais non disponibles en N-1 et enfin mettra à jour les autres paramètres pour arriver à la tarification N.

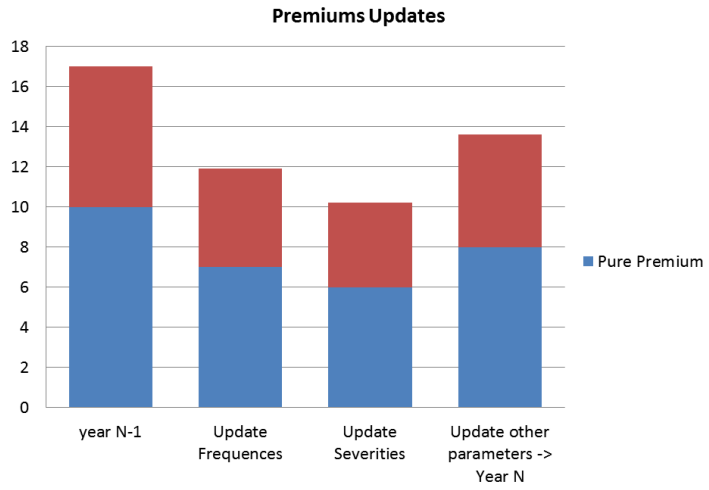

FIGURE 6 – exemple : Update Fréquences-Coûts quand il y a une seule loi de générateur Atypique

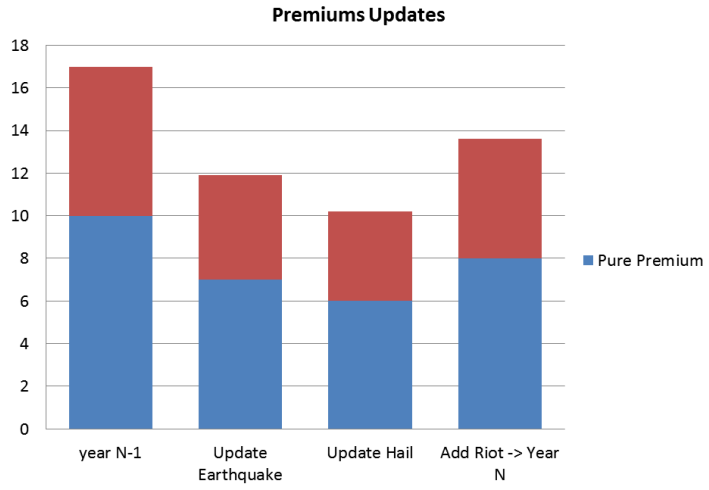

FIGURE 7 – exemple : Update des périls CAT

Cette module est simple à implémenter. Il suffit de distinguer le cas CAT/Atypique, les différents scénarios de mise à jour (en utilsant les conditions If...Else...) et dans chaque scénario, boucler les fonctions CAT_Modeling_func() (si modélisation CAT) ou NON_CAT_Modeling_func() (si modélisation atypique) et XS_Application_func() et éventuellement XSA_Application_func() (si appliquer XS Aggregate).

3.3 Contribution de modèles vendeurs

Dans le cas de la modélisation CAT, pour un péril donné, nous disposons potentiellement plusieurs visions du risque. Nous voulons comparer les primes pures et primes techniques suivant les différentes visions (3 modèles vendeurs M1, M2, M3 + vision groupe M4 + vision de l'utilisateur User's Blending). Pour ce fait, j'ai développé un module Contribution des modèles vendeurs.

Dans ce module, Hermès va remplir successivement le tableau des coeffcients de blending à 100% d'un des modèle vendeurs (M1, M2, M3) et enfin à 100% du M4 et faire la modélisation des pertes et la tarification correspondante (en appelant la fonction CAT_Modeling_func(), XS_Application_func() et XSA_Aplication_func() si appliquer XS Aggregate) pour obtenir la prime pure et prime technique dans chacun des 4 cas. Notons-nous que tous les modèles ne sont pas toujours disponibles pour un péril considéré. Dans le cas où un modèle vendeur M_i (i=1,2 ou 3) est manquant, nous prenons 100% M4 plutôt que 100%M_i.

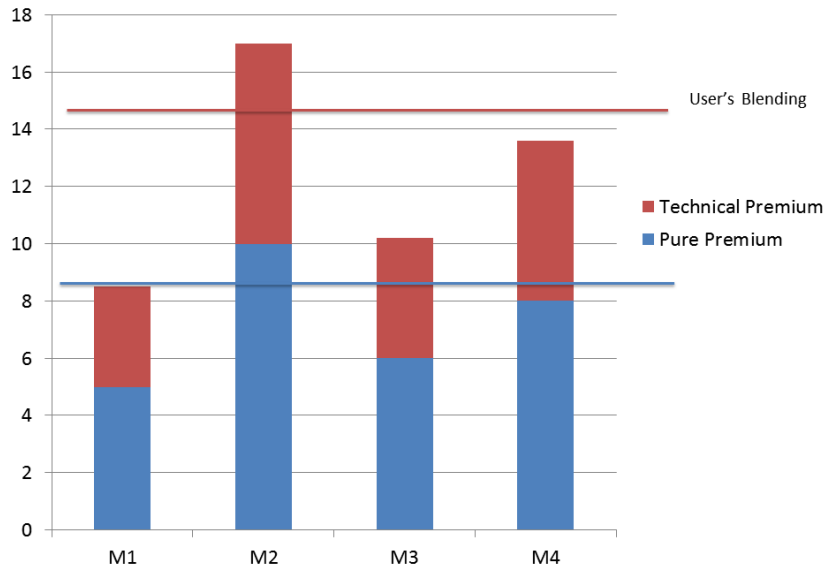

FIGURE 8 – exemple : Contribution des modèles vendeurs

Dans l'exemple ci-dessus, nous pouvons constater une forte différence entre les visions de modèles vendeurs. La vision M4 retenue par le groupe est une moyenne pondérée des trois modèles vendeurs. Le blending choisi par l'utilisateur n'est pas le même que celui choisi par le groupe (ce qui se traduit par un niveau de prime un peu plus faible).

4 Backtesting

Une fois développée les nouvelles fonctionnalités, j'ai re-testé Hermès avec plusieurs générateurs de pertes et plusieurs programmes de réassurance pour voir s'il y a des bugs et si la nouvelle version donne (plus ou moins) des mêmes résultats que l'ancienne version. Les petits bugs (liés souvent au format et à la dimension des objets : matrice/vecteur/data frame) ont été identifiés et corrigés rapidement. Cependant j'ai constaté un problème dans la modélisation CAT quand l'utilisateur fait du blending. Après avoir effectué plusieurs tests, j'ai remarqué que si au moins deux parmi trois modèles vendeurs interviennent dans le blending, les résultats sont cohérents alors que si on prend 100% d'un modèle vendeur, les résultats sont très différents entre les deux versions Hermès. J'ai ensuite essayé de faire un blending 100% M1 et un blending 99,99% M1 + 0,01% M2. Les résultats des deux cas sont quasiment les mêmes dans la nouvelle version d'Hermès. Le résultat obtenu avec 99,99% M1 + 0,01% M2 de l'ancienne version d'Hermès s'approche aussi des résultats précédents. Pourtant le résultat obtenu avec 100% M1 dans l'ancienne version d'Hermès s'éloigne beaucoup tous les trois autres résultats.

L'explication, après revoir le code Hermès de l'ancienne version d'Hermès, est que cette version raisonne de façon différente si l'utilisateur prend 100% d'un modèle vendeur : les pertes seront modélisées directement suivant le générateur de ce modèle. Le raisonnement de la nouvelle version d'Hermès (celui que nous allons retenir dorénavant) est comme suivant : on commence toujours par modéliser les pertes suivant la vision M4 et on les corrige pour avoir les OEP cibles donnés par la vision M1. D'où l'écart constaté en résultats.

5 Limites et perspectives de la nouvelle version d'Hermès

Dans cette dernière partie je présenterai les limites de mon travail et mes perspectives sur la nouvelle version d'Hermès. Avant de rentrer en détail, je vais présenter le schéma final d'Hermès après mes modifications. J'ai restructuré les codes Hermès par bloc (suivant la tâche), des fois par fonction pour pouvoir les réutiliser facilement.

Avec une durée assez courte pour ce stage, je n'ai pas eu le temps pour améliorer certaines limites que j'ai pu constater.

- les phases de mise à jours sont mises dans Common Core, que Hermès va boucler autant de fois que le nombre de programme sélectionné par l'utilisateur. Pour l'instant le nombre de programme est fixé à un par défaut donc il n'a aucun problème. Pourtant si nous envisageons plusieurs programmes à tarifer par Hermès, nous perderons un peu du temps de calcul inutile parce que pour chaque programme de réassurance, on remodélise les pertes dans le module de mise à jour. En fait, il suffit de modéliser ces pertes là une fois pour tout et après appliquer les différents programms à ces pertes. J'aurais dû séparer chaque phase de mise à jour en deux parties distinctes (modélisation des pertes et application du programme de réassurance) comme dans la modélisation year N et boucler seulement la parties d'application du programme de réassurance.
- Je n'ai pas eu le temps pour modifier la façon dont Hermès modélise la cadence de paiement à partir du fichier générateur. Il faut détecter quel type de cadence est présent (la cadence directe ou il faut modéliser par la méthode déterministe avec les pourcentages cibles) et faire des modifications dans

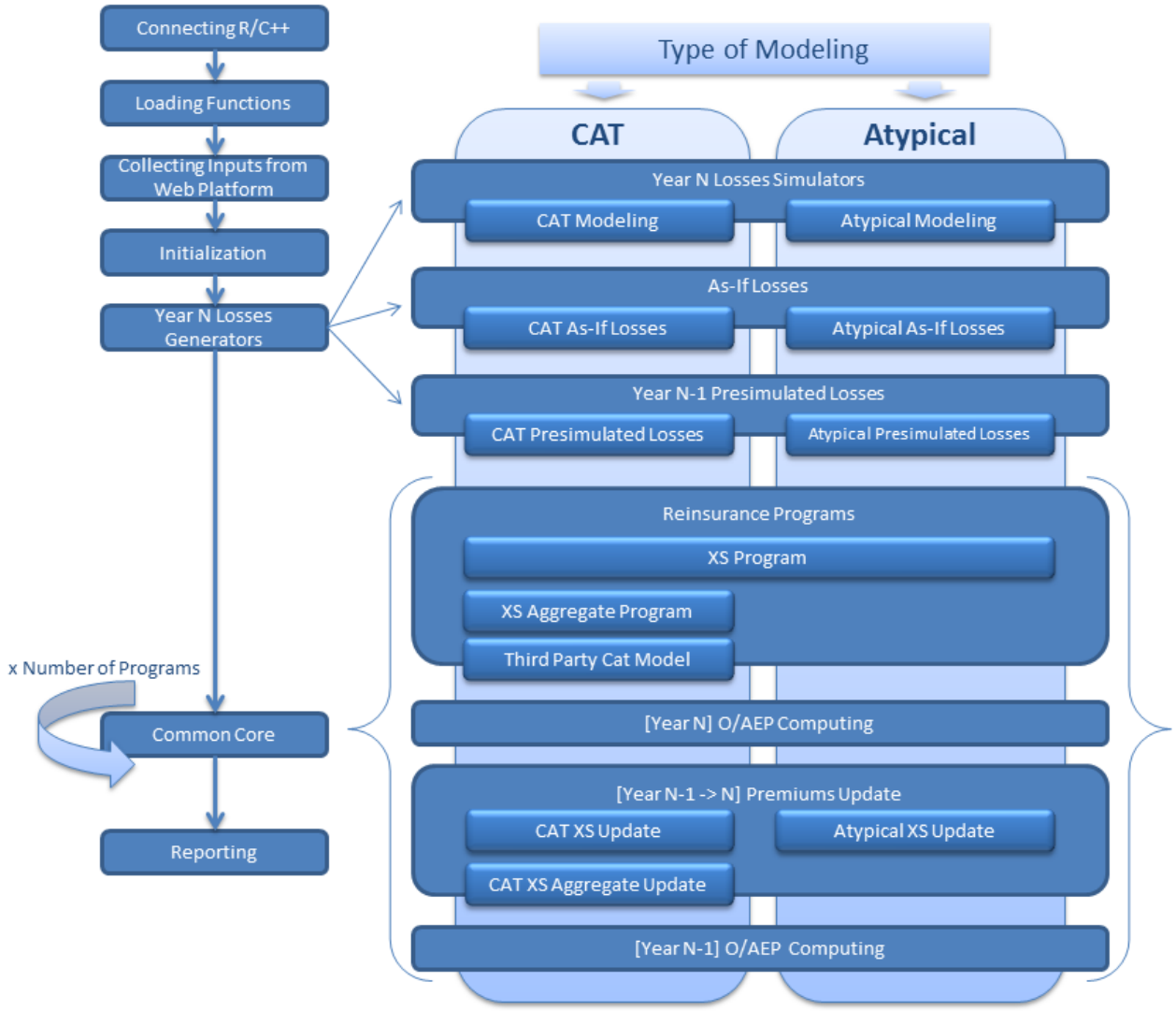

FIGURE 9 – Schéma Hermès

la partie où Hermès génère la matrice de cadencement de paiements.

– Pour l'instant, Hermès passe toujours par une première étape de compilation des fonctions C++ à chaque fois qu'on relance R. On pourrait envisager la pré-compilation de ces fonctions.

Ce sont des points essentiels à corriger pour rendre Hermès encore plus puissant. Je n'ai pas pu rester jusqu'à la fin du projet pour l'intégration d'Hermès dans la plateforme web. Je n'ai pas pu non plus développer les autres méthodes de tarification (courbe d'exposition, méthode historique, etc.). Je n'ai défini que les inputs/outputs pour chacun des modules correspondant à ces méthodes pour ce ça soit cohérent avec la nouvelle structure Hermès. Toutefois, selon les tests que j'ai effectués, je peux conclure

que Hermès version R/C++ peut générer les pertes plus rapidement que le temps mis pour lire le vecteurs des pertes et des cadences de paiements dans les fichier csv externes. Donc, la méthode historique (avec les pertes pré-simulées et stockées dans les fichiers csv) sera potentiellemnt plus longue que la méthode de simulation directe des pertes et d'ailleurs nécessitera beaucoup d'espace de stockage sur le réseau.

Annexes

A1. Récapitulatif des fonctions C++ intervenant au code Hermès

Dans la modélisation des pertes Atypiques (fonction NON_CAT_Modeling_func()) :
- cumulation_CPP()
- Payment_Pattern_Calculation.mean_DH() :
  - sum_per_row_CPP()
  - percentage_payment_CPP()

Dans la tarification en appliquant les traités XS (fonction XS_Application_func()) :
- sum_per_year_CPP()
- discount_rate_appli_CPP()
- Layer_cost_DH()
  - limit_priority_with_index_clause_appli_CPP()
  - invcumsum2_CPP()
  - AAD_AAL_Appli()
- Layer_price2_DH()
  - sum_per_year_CPP()
  - discount_rate_appli_CPP()
  - division_CPP()
  - exhaustion_prob_CPP()
  - AAD_AAL_Appli()

Dans le script où les OEP/AEP sont calculés (script OEP_Computing) :
- max_per_year_CPP()
- sum_per_year_CPP()
- sum_per_row_CPP()
- OAEP_Gen_function_CPP() :
  - max_per_year_CPP()
  - sum_per_year_CPP()

A2. Liste des abréviations

1. P&C : Property & Casualty (IARD et Responsabilité Civile)
2. QS : Quote Share (QP : Quote Part)
3. XL : Excess of Loss (XS : Excédent de Sinistre)
4. AAD : Annual Aggregate Deductible (franchise globale)
5. AAL : Annual Aggregate Limit (limite globale)
6. OEP : Occurrence Exceedance Probability
7. AEP : Aggregate Exceedance Probability