NAM

Summary and discussion of software benchmarking for Groningen PSHRA code

Stephen Bourne ad Steve Oates

Datum June 2016

Editors Jan van Elk & Dirk Doornhof

General Introduction

A probabilistic seismic hazard and risk assessment (PSHRA) tools has been developed for Groningen induced seismicity. The PSHRA workflow is based on Monte Carlo sampling of a set of underlying distributions describing earthquake occurrence (the seismological model), ground motion calculations (GMPEs), building damage (fragility functions) and consequences thereof (injury functions). Calculated results are aggregated to give a probabilistic summary of the induced seismic hazard and risk.

Many of the methods implemented are novel, little research having been done previously on Monte Carlo approaches to the problem of time-varying induced seismicity. This report described the assurance procedure for the development of the hazard and risk assessment tool, based on implement using two different computer codes (one in Python and the other in the C language) and comparison of the results.

This approach to cross-validation, whereby two separate codes with the same essential functionality have been developed in parallel and their outputs compared throughout the development cycle, has repeatedly proved its worth as a means of identifying and resolving bugs and inconsistencies. This assurance of the implementation for the Groningen PSHA codes exceeds the practice for critical facilities.

NAM

Title	**Summary and discussion of software benchmarking for Groningen PSHRA code**		**Date**	June 2016
			Initiator	NAM
Author(s)	Stephen Bourne & Steve Oates	**Editor**	Jan van Elk Dirk Doornhof	
Organisation	NAM	**Organisation**	NAM	
Place in the Study and Data Acquisition Plan	A probabilistic seismic hazard and risk assessment (PSHRA) tools has been developed for Groningen induced seismicity. The PSHRA workflow is based on Monte Carlo sampling of a set of underlying distributions describing earthquake occurrence (the seismological model), ground motion calculations (GMPEs), building damage (fragility functions) and consequences thereof (injury functions). Calculated results are aggregated to give a probabilistic summary of the induced seismic hazard and risk. Many of the methods implemented are novel, little research having been done previously on Monte Carlo approaches to the problem of time-varying induced seismicity. This report described the assurance procedure for the development of the hazard and risk assessment tool, based on implement using two different computer codes (one in Python and the other in the C language) and comparison of the results. This approach to cross-validation, whereby two separate codes with the same essential functionality have been developed in parallel and their outputs compared throughout the development cycle, has repeatedly proved its worth as a means of identifying and resolving bugs and inconsistencies. This assurance of the implementation for the Groningen PSHA codes exceeds the practice for critical facilities.			
Directly linked research	1. Interim Hazard and Risk Assessment induced Seismicity Groningen, NAM, Nov. 2015, 2. Winningsplan 2016, NAM, April 2016.			
Used data				
Associated organisation				
Assurance	This research describes assurance procedures used in building the hazard and risk assessment tools.			

Summary and discussion of software benchmarking for Groningen PSHRA code

Contents

Introduction 3
Checking the implementation of input models 4
Benchmarking code results 4
Ground-motion (GMPE) benchmark 5
Hazard benchmarks 6
Risk benchmarks 9
Discussion of example benchmarks 11
Future code deployment 13
Concluding comments 13
References 14

Introduction

A significant effort has gone into developing and applying probabilistic seismic hazard and risk assessment (PSHRA) tools for Groningen induced seismicity. See [1], [2], [3], [4] and [5] for details of the methods implemented and a summary of the status of the work at the time of writing. The PSHRA workflow has been founded on Monte Carlo sampling of a set of underlying distributions describing earthquake occurrence (the seismological model), ground motion calculations (GMPEs), building damage (fragility functions) and consequences thereof (injury functions). Calculated results are aggregated to give standard types of probabilistic output summarizing the induced seismic hazard and risk.

Many of the methods implemented are novel, little research having been done previously on Monte Carlo approaches to the problem of time-varying induced seismicity. From the outset, we have developed and maintained two distinct computer codes with essentially the same PSHRA functionality – one written in Python and the other in C. Throughout the development process and ongoing cycle of hazard and risk assessments, the outputs from these two codes have been compared against each other as a means of validating our results. This report describes the benchmark tests applied to cross-validate our PSHRA codes and summarises the vintage of results which support the 2016 Winningsplan submission.

This approach to cross-validation, whereby two separate codes with the same essential functionality have been developed in parallel and their outputs compared throughout the development cycle, has repeatedly proved its worth as a means of identifying and resolving bugs and inconsistencies. In PSHA practice for critical facilities such as nuclear power plants, it is generally a requirement that the hazard calculation codes undergo formal qualification. This typically requires a software test plan and related acceptance criteria to be determined based on the safety function of the software. The results of the test cases specified in the plan are compared to the corresponding anticipated output. The anticipated output can be hand calculations, or calculations using alternative software (typically Excel, or other previously qualified software). The comparison between software results and anticipated output needs to demonstrate that the differences are within the acceptance criteria specified in the test plan. However, other than for very simple input configurations, an estimation of the anticipated output is typically difficult to do except by comparison with the results from other codes, which may give significantly different results [6]. The assurance of the implementation for the Groningen PSHA codes therefore exceeds the practice for critical facilities.

With a suitably-qualified code, which means that the program is performing the intended calculations for a given input, checks are also required that the input parameters of the models are being correctly entered. For some elements this can be done through reproduction of exact results (which can be done, for example, for the ground-motion prediction equations), but for the full hazard integrations this is more difficult. In one nuclear project, the full logic-tree was implemented in two separate calculation codes, each of which had been previously qualified [7], but this was an exceptionally rigorous approach compared to standard practice. To develop separate implementations of both the calculation engine

and the input parameters, as has been implemented for the Groningen PSHA, is without precedent in this field.

The main reasons for developing two distinct Monte Carlo PSHRA codes in this way can be summarized as follows.

- Benchmarking: allows for independent cross-checking of the outputs.
- Debugging: helps in the identification and fixing of bugs.
- Algorithmic differences in implementations lead to a greater understanding of methods and their sensitivities and help identify more robust solutions.
- Different codes have different characteristics: Python generally leads to faster code development but slower run times of the finished code compared with C; these computational speed differences can be of the order of factors of 10 or 100. However, the slower code development time for C means there is an unavoidable delay between enhancements to the hazard and risk models, and the issue of validated results.
- Python has publications-grade plotting functionality (Matplotlib) and a vast array of scientific analysis tools (Scipy & Numpy) coded, benchmarked and maintained by a global community of open-source developers. C, on the other hand, lacks most such facilities and so encourages a minimalist approach to programing in which alternatives to complex, often CPU-intensive steps, are used where possible.

Checking the implementation of input models

The first step of the quality control process applied to the Groningen PSHRA involves checking the implementation of input models, and this is done separately in both of the Monte Carlo PSHRA codes. At the same time that the PSHRA code developers are provided with a given version of the seismological, GMPE, fragility and consequence models, example calculations and results are also provided, so that the implementation of the models can be checked. For example, in the case of the fragility models, the probability of collapse under different levels of ground shaking for a number of different building typologies is provided. For the GMPEs, plots of predicted median response spectra for specific scenarios and curves of median predicted accelerations as a function of magnitude and distance are generated using different software from that used for the PSHRA codes (for example, Excel or Matlab). The risk engine developers begin by reproducing these plots from their own implementations.

Benchmarking code results

The Python and C codes developed differ not only in the programing language chosen but also in the details of the algorithms used. The two codes share however the same basic inputs defining the seismological model, the ground motion calculations, the database of buildings in the area, the building fragility functions and the consequence model for the eventual effects of the surface ground motion.

At key stages during the ongoing development of hazard and risk assessments, results generated by the Python and C codes have been checked against each other. Calculations of synthetic earthquake

catalogues, from the shared input files defining the seismological model, have been shown to give near-identical distributions of numbers of earthquakes above given magnitude thresholds. A selection of plots from recent benchmark comparisons of the ground motion, hazard and risk calculations, using the suite of inputs used to generate the results for the April 1st 2016 Winningsplan submission (V2.5) [5], are shown in the following pages.

Ground-motion (GMPE) benchmark

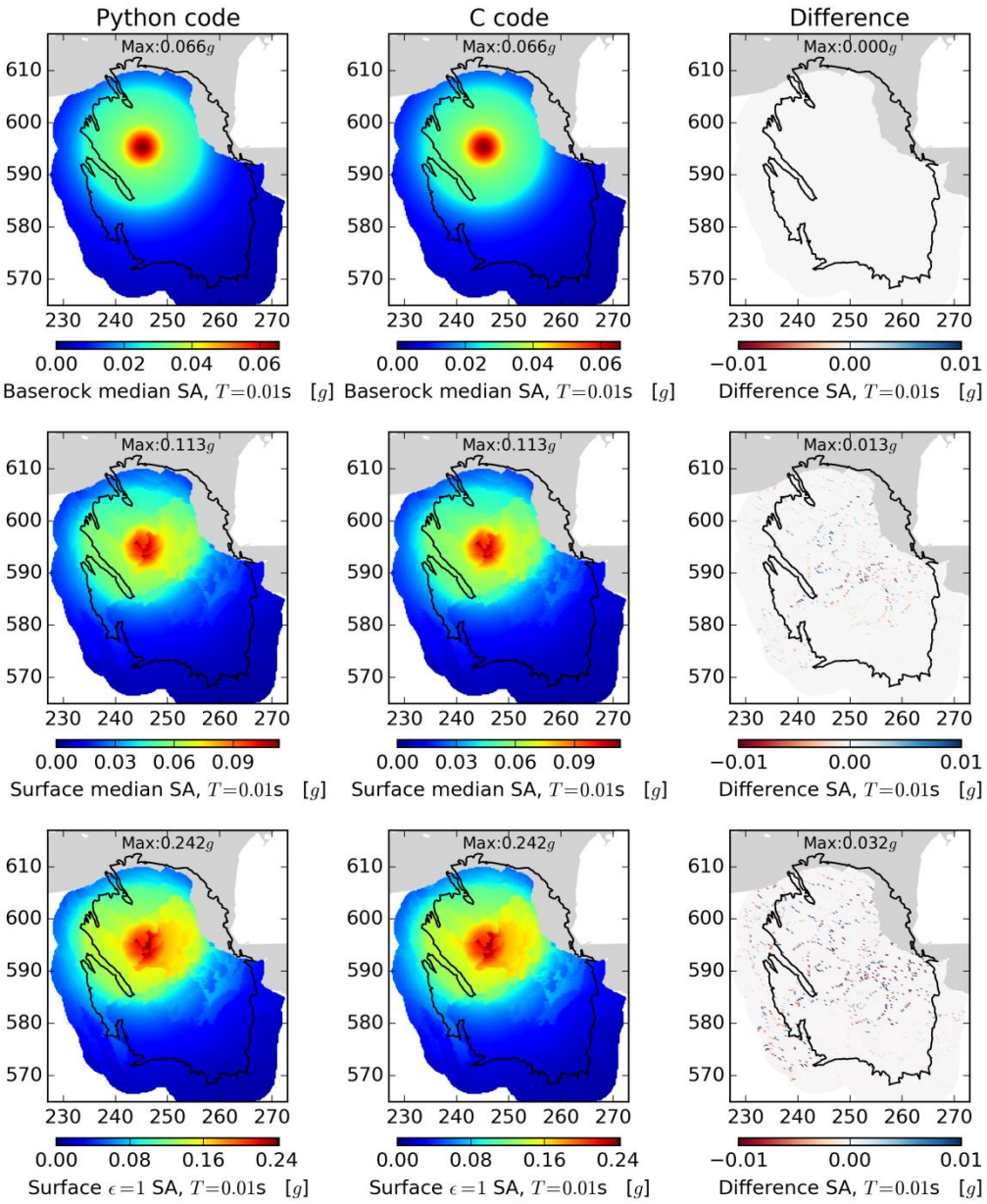

Figure 1 *Ground motion computed on a 250m regular grid for a single M = 5 event at (245,595,3) km. Compares base rock (ε=0), surface (ε=0), surface (ε=1). The few discrepancies are generally small and related to the different spatial discretization schemes used.*

Hazard benchmarks

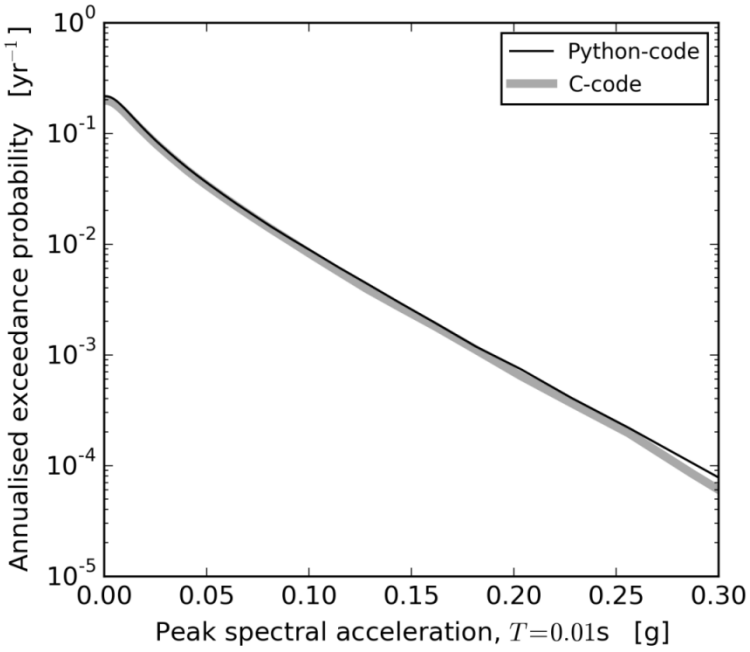

Figure 2 *Hazard curve calculation at (X=245, Y=592 – a location close to Loppersum) for 33 bcm per year production scenario, for a five year simulation period, 2016-2021.*

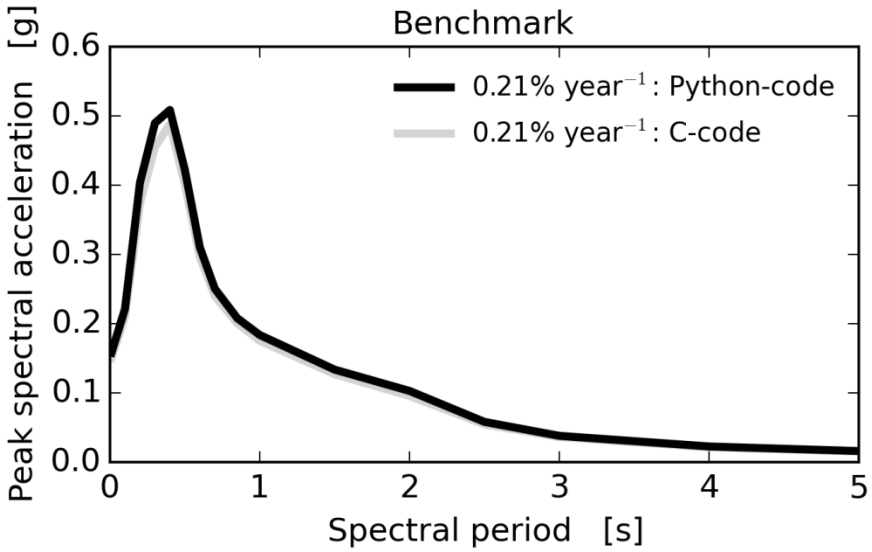

Figure 3 *Uniform hazard spectrum at (X=245, Y=592) for the base case of the logic tree (Mmax=5.75 and central branch of the GMPE) for 33 bcm per year production scenario, for a five year simulation period, 2016-2021, 0.2%/year chance of exceedance.*

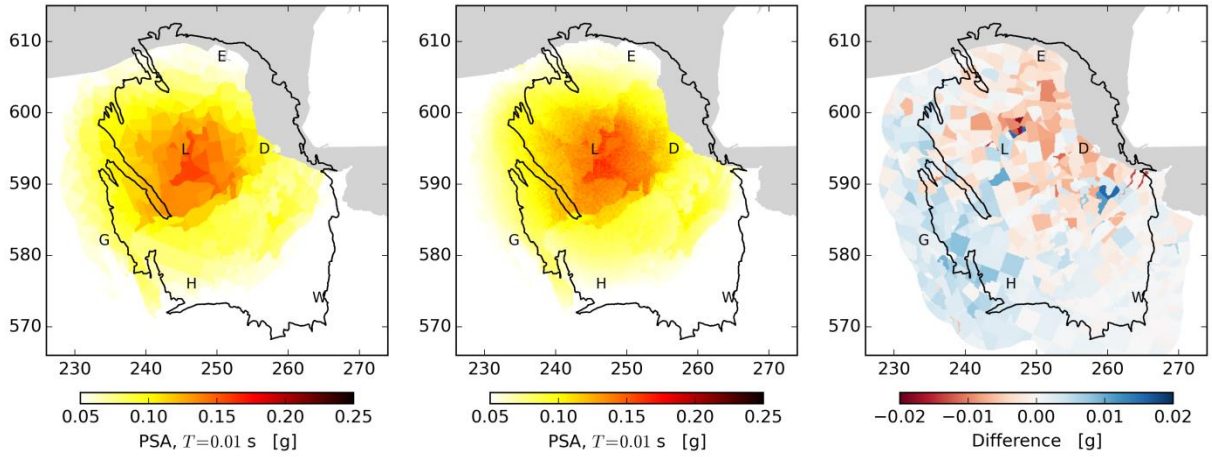

Figure 4 *Hazard maps for the base case of the logic tree (Mmax=5.75 and central branch of the GMPE) for 33 bcm per year production scenario, for a five year simulation period, 2016-2021, 0.2%/year chance of exceedance.*

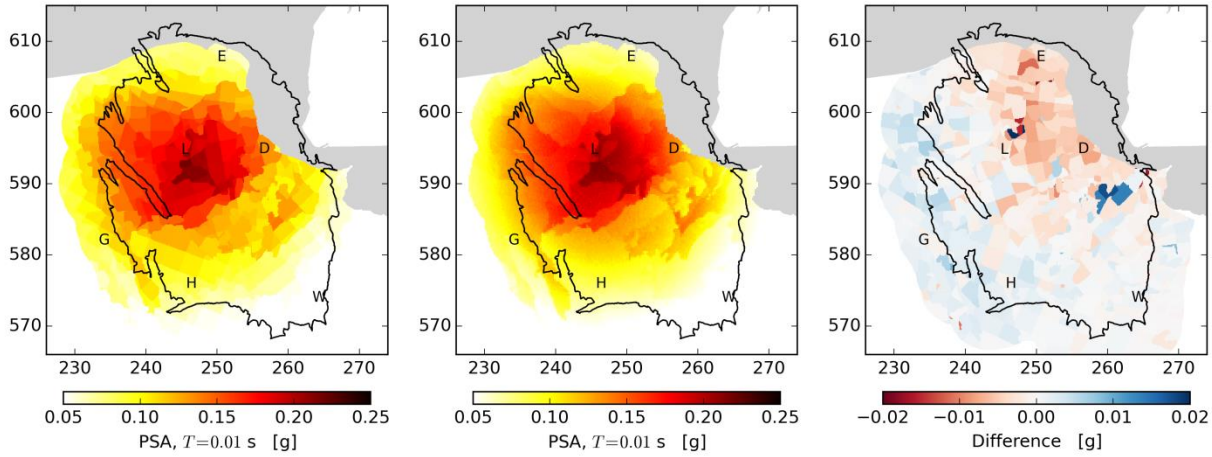

Figure 5 *Hazard maps for the logic tree mean (Mmax=5.0, 5.75 & 6.50 for central and upper branches of the GMPE with V2.5 logic tree weights) for 33 bcm per year production scenario, for a five year simulation period, 2016-2021, 0.2%/year chance of exceedance.*

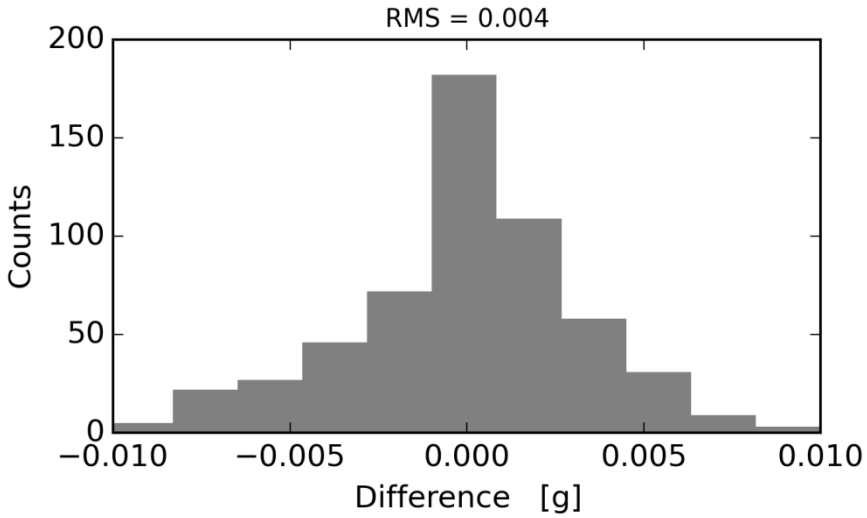

Figure 6 *Histogram of differences between Python and C hazard maps, calculated at the regular output grid locations used by the C code. RMS difference is 0.004g; fractional difference: c. 0.6%. For the base case of the logic tree (Mmax=5.75 and central branch of the GMPE) for 33 bcm per year production scenario, for a five year simulation period, 2016-2021, 0.2%/year chance of exceedance.*

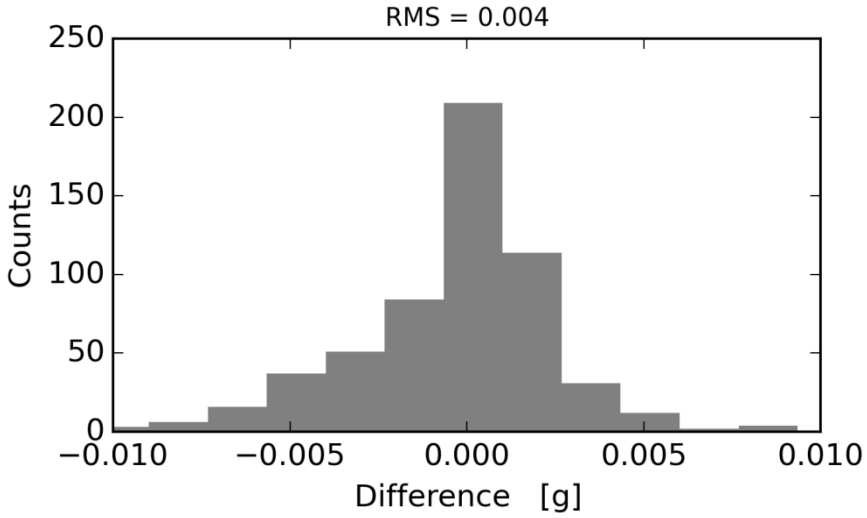

Figure 7 *Histogram of differences between Python and C hazard maps, calculated at the regular output grid locations used by the C code. RMS difference is 0.004g; fractional difference: c. 0.6%. For the logic tree mean (Mmax=5.0, 5.75 & 6.50 for central and upper branches of the GMPE with V2.5 logic tree weights) for 33 bcm per year production scenario, for a five year simulation period, 2016-2021, 0.2%/year chance of exceedance.*

Risk benchmarks

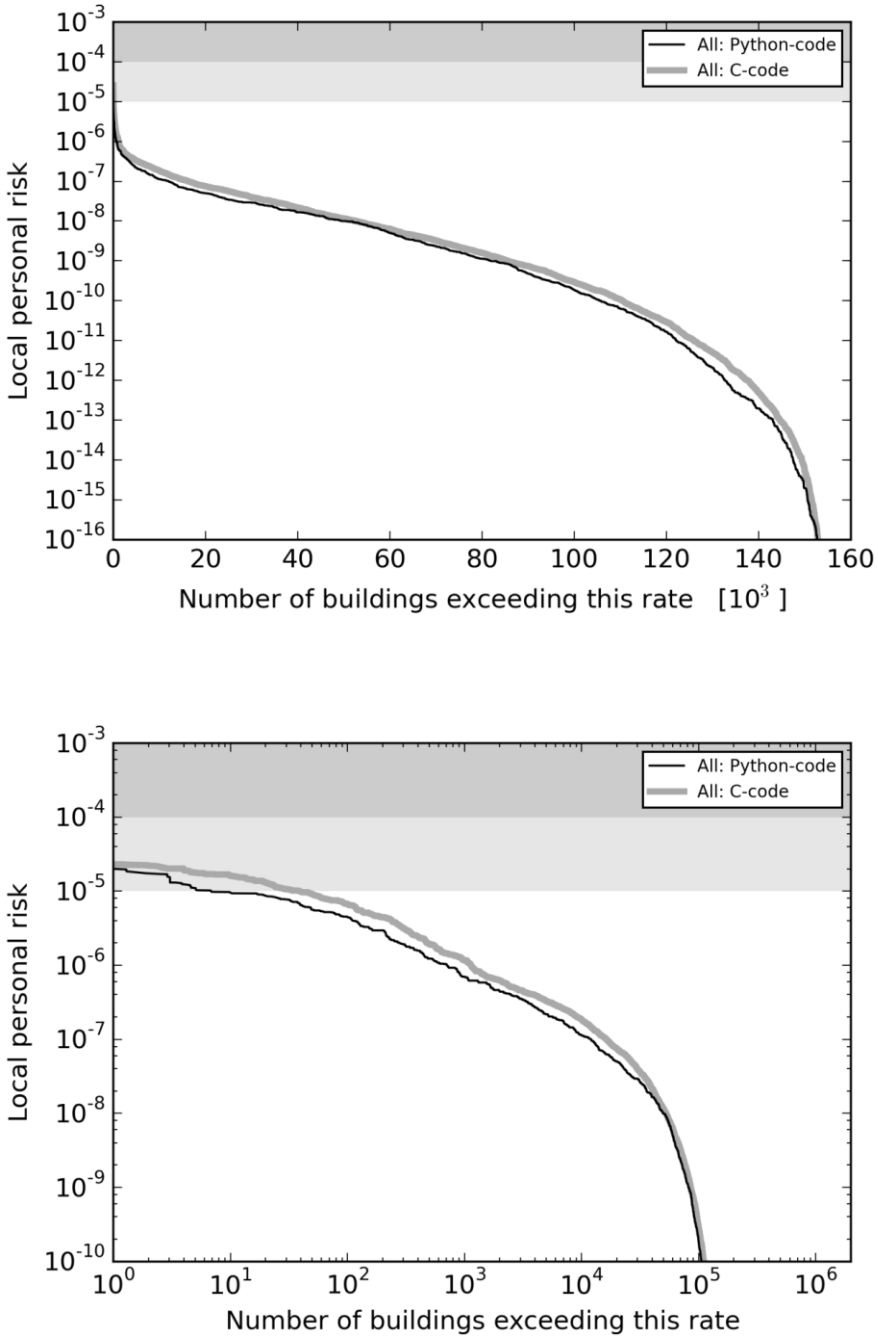

Figure 8 *Plots of inside local personal risk (ILPR), for all building typologies, for the base case of the logic tree (Mmax=5.75 and central branch of the GMPE) for 27 bcm per year production scenario, for a five year simulation period, 2016-2021. Linear and logarithmic horizontal axes are used to emphasise differences over different ranges.*

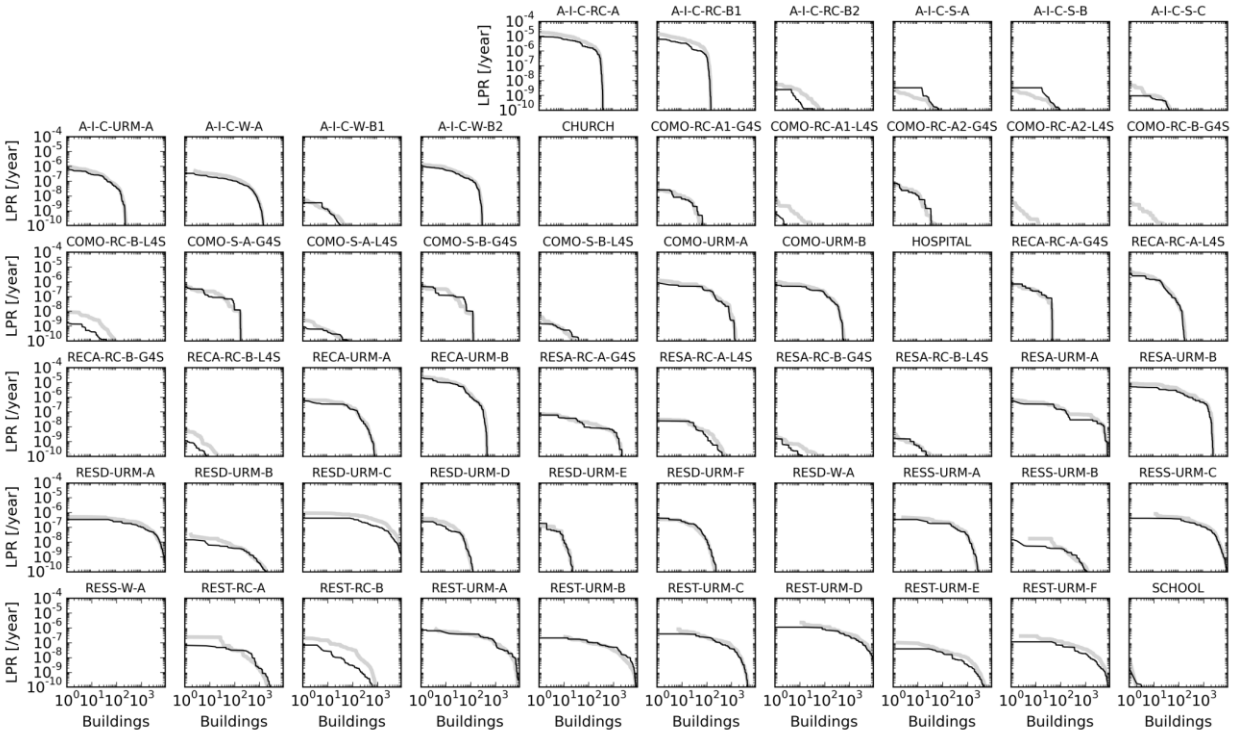

Figure 9 *Plots of inside local personal risk (ILPR), for each individual building typology, for the base case of the logic tree (Mmax=5.75 and central branch of the GMPE) for 27 bcm per year production scenario, for a five year simulation period, 2016-2021. Dark curves correspond to the Python implementation, grey curves to the C code.*

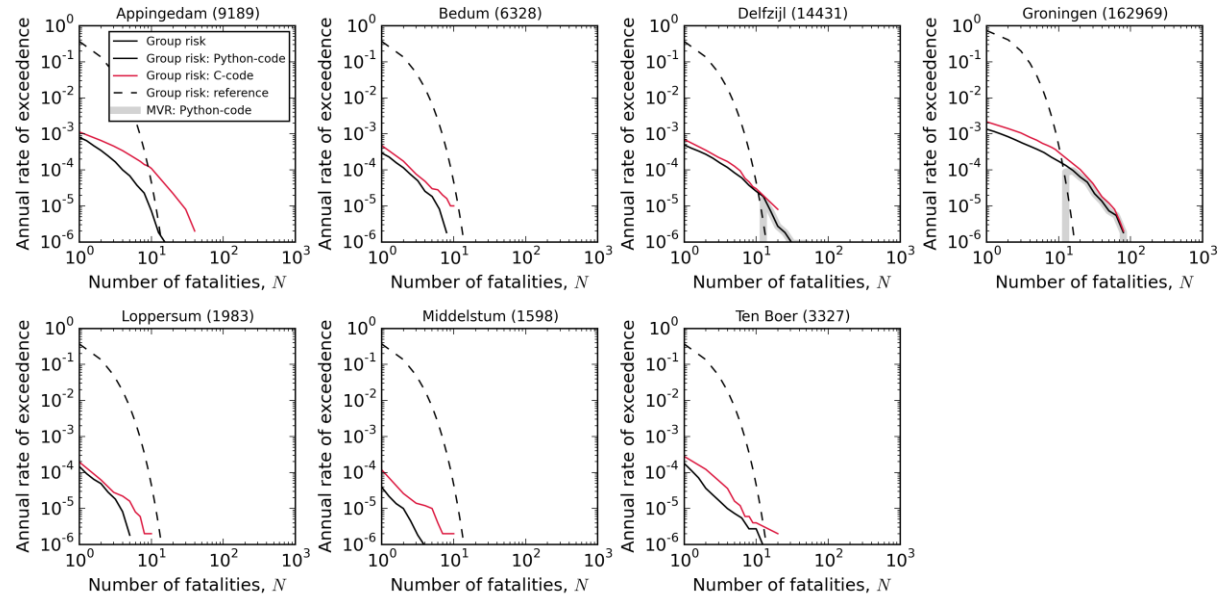

Figure 10 *Comparison of Python and C code FN curves (base case of the logic tree: Mmax=5.75 and central branch of the GMPE; 27 bcm per year production scenario, 2016-2021) for seven communities: Appingedam, Bedum, Delfzijl, Groningen, Loppersum, Middelstum and Ten Boer. These FN curves show the estimated annual rate of occurrence (F) of individual earthquake events leading to exceedance of the stated number of fatalities (N). That is, the value on the vertical axis is the cumulative frequency of experiencing N or more fatalities. The*

communities comprise all buildings within the boundaries given by CBS. The numbers in brackets are the inside populations, averaged over day and night.

Discussion of example benchmarks

Referring to the ground motion and hazard calculations, notice the generally very close agreement between the outputs from the Python and C codes. The main differences between the maps of surface ground motion and hazard generated with Python and C (Figures 1, 4 and 5) seem to be due to the different discretization choices made to represent the near-surface zonation.

In the Python code, an exact representation of the identified boundaries of the individual zones of different surface soil characteristics has been used to define an irregular output grid for the ground motion calculation (as shown in figure 11). This has been achieved using a Scipy percolation function (scipy.ndimage.measurements) which detects regions of identical properties within a boundary.

In the C code, the near-surface characteristics, originally mapped on a 100mx100m grid, have been upscaled onto a regular but coarser grid. Where the surface geology zonation boundaries cut across a grid cell, average properties are calculated by resampling the distribution of soil types falling within that output cell and assigning the selected soil type to the whole cell for the catalogue being simulated. This resampling process is repeated for each new simulated catalogue.

Tests have shown that if the C code's grid is coarser than that on which the surface zonation has been defined, the zone boundaries are effectively smoothed by the resampling process but that the averaged properties which contribute to the probabilistic hazard calculation agree well with the Python code output. This can be seen clearly in figures 1, 4 and 5 where the differences between the results from the Python and C codes are seen to lie mainly along the boundaries between zones.

We were driven by practical considerations to follow these two different approaches to the near-surface discretization which, on reflection, we realized had complementary merits. The availability of the Python open-source percolation function enabled the exact representation of mapped zones of distinct properties to be honoured in the Python code without significant additional coding. This however was not the case in C. Rather than spend significant effort coding and testing an equivalent C percolation function, it was decided to exploit the speed advantage of the C-code to work on a grid which was regular but significantly finer than that used by the Python code. The regular grid approach necessarily smoothes the near-surface zonation but an ergodic argument would suggest that this may not bias the hazard and risk results which are averages over a large number of realisations. Moreover, working with a finer calculation grid will result in denser sampling of the base rock ground motion field which should result in correspondingly more accurate representation of lateral variations in the base-rock seismic hazard. However, another consequence of the finer calculation grid is that the approximation to modelling spatial correlation of ground motions through the use of a larger grid is effectively lost. This is only of consequence for aggregated risk metrics but not for local personal risk (LPR).

A second algorithmic difference between the C and Python implementations concerns the way in which aftershocks are treated. The ETAS – Epidemic Type After Shock – model, which details spatial and temporal earthquake clustering, is an important part of the seismological model [4] used by both codes.

This model gives a means of calculating the distributions of aftershock times and locations, relative to the time and location of a main shock which is considered to be the parent earthquake of the aftershock sequence. The structure of the loop on all events used in the C code motivated an approximate treatment of higher aftershock generations which differs from the Python implementation's exact treatment. In the C code, although the numbers of aftershocks and their magnitude distribution are calculated as specified by the ETAS model, honouring the properties of higher generations of aftershocks, the locations and times of aftershocks are not referred to those of their true parents. They are instead calculated relative to those of the original main shock which gave rise to the first generation of aftershocks. This approximation was chosen to avoid the need for storing and repeatedly referring back to the properties of all aftershocks across all generations. It is recognized that it will result in somewhat denser clustering of groups of aftershocks but it is expected that higher generations of aftershocks, and this approximation in particular, will make relatively small contributions to the overall results. Although it is to be expected that this approximation will tend to increase the localization of the hazard, there is no clearly discernible evidence of such an effect in the comparisons shown here. Indeed, the structures in the difference maps shown for ground motion and hazard, clearly follow the surface zonation boundaries strongly suggesting that the surface discretization difference already discussed is dominant.

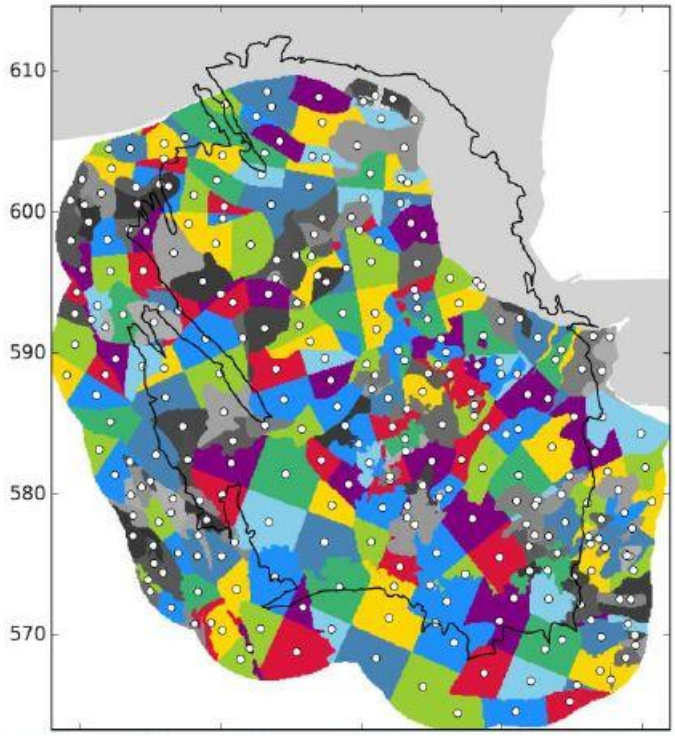

Figure 11 *Map showing zone boundaries and centroids generated by projecting near surface properties onto an irregular grid using the Python percolation function. This approach has been shown to honour nearly all the structure in the densely sampled input map. An alternative probabilistic resampling onto a regular grid was implemented in the C code.*

Future code deployment

With the completion of the PSHA assessment for Winningplan 2016, there is an opportunity for professional software developers to build a single stable code platform from which a tool can be developed that will be used for regular updates of the PSHA. Recognizing the requirement that the speed of the deployed code should be comparable to that of the C code, alternative approaches to coding, which combine C and Python, are being explored. As well as expected improvements in the usability and robustness of the code, having a professionally produced code base will enable a more rigorous approach to testing and debugging. It is intended that some of the methods of Test Driven Development, such as unit testing [8], will be used. An example of this approach is the OpenQuake-engine [9], the open-source software for probabilistic hazard and risk assessment developed by the Global Earthquake Model group based in Pavia.

This code deployment effort is currently in its start-up phase. Final scope of this effort will be decided by NAM, informed by the initial exploration of our PSHRA codes by the software team. The current intention is that a first product will become available towards the end of 2016 but this schedule needs to be confirmed.

Concluding comments

The benchmark calculation results shown here compare the outputs from the Python and C implementations of our Monte Carlo PSHRA methodology. We see excellent agreement of the ground motion and hazard outputs and very good agreement for the risk outputs although there are some clear detailed differences to be seen when comparing results for individual building typologies. Further work is ongoing to identify the origin of these remaining differences and then make any necessary upgrades to the software.

The independent cross-validation approach described here has proved invaluable in the code development cycle, both for identifying and fixing bugs but also for improving our understanding of the algorithm used and their sensitivities. This ongoing benchmarking process will continue to be an important element of our approach as we further develop the probabilistic hazard and risk workflow.

Recent experience has demonstrated the importance of completing the cross-validation exercise before releasing a batch of hazard and risk results. The Groningen seismic hazard and risk models are very complex with a large number of variables and correlation structures, making rigorous cross-validation essential. Schedules that lead to insufficient time between the development of new inputs (seismological model, ground-motion prediction equations, site response zonations, fragility and consequence functions) and the delivery of results inevitably create the danger of inaccurate hazard and risk estimates subject to subsequent correction.

With a lower pace of scientific development of the PSHA methodology, deployment of our hazard and risk engine by software professionals in a stable, maintainable environment is an important new development. It is hoped that this will increase the robustness of our code and make the workflow easily accessible to a wider user community.

References

1. Bourne, S.J., Oates, S.J., van Elk, J., Doornhof, D., 2014. A seismological model for earthquakes induced by fluid extraction from a subsurface reservoir. J. Geophys. Res. Solid Earth 119.
2. S. J. Bourne, S. J. Oates, J. J. Bommer, B. Dost, J. van Elk, and D. Doornhof. A Monte Carlo Method for Probabilistic Hazard Assessment of Induced Seismicity due to Conventional Natural Gas Production. Bulletin of the Seismological Society of America, Vol. 105, No. 3, pp. 1721–1738, June 2015
3. Bourne, S.J., Oates, S.J., 2014. An activity rate model of induced seismicity within the Groningen Field. Technical Report. Shell Global Solutions International. Rijswijk, The Netherlands.
4. Bourne, S.J., Oates, S.J., 22 June 2015. An activity rate model of seismicity induced by reservoir compaction and fault reactivation in the Groningen gas field. CONFIDENTIAL DRAFT
5. NAM. Jan van Elk et al. Technical Addendum to the Winningsplan Groningen 2016. Production, Subsidence, Induced Earthquakes and Seismic Hazard and Risk Assessment in the Groningen Field.
6. Thomas, P., I. Wong & N. Abrahamson (2010*). Verification of probabilistic seismic hazard analysis computer programs*. PEER Report 2010/106, Pacific Earthquake Engineering Research Center, UC Berkeley, California.
7. Bommer, J.J., F.O. Strasser, M. Pagani & D. Monelli (2013). Quality assurance for logic-tree implementation in probabilistic seismic hazard analysis for nuclear applications: A practical example. *Seismological Research Letters* **86**(6), 938-945.
8. https://en.wikipedia.org/wiki/Test-driven_development and https://en.wikipedia.org/wiki/Unit_testing
9. Pagani, M., Monelli, D., Weatherill, G. A. and Garcia, J. (2014). *The OpenQuake-engine Book: Hazard*. Global Earthquake Model (GEM) Technical Report 2014-08, doi: 10.13117/-GEM.OPENQUAKE.TR2014.08, 67 pages.