Redesign of the GATE PET coincidence sorter

2016 Phys. Med. Biol. 61 N522

(http://iopscience.iop.org/0031-9155/61/18/N522)

IOP Publishing | Institute of Physics and Engineering in Medicine Physics in Medicine & Biology

Phys. Med. Biol. **61** (2016) N522–N531 doi:10.1088/0031-9155/61/18/N522

Note

Redesign of the GATE PET coincidence sorter

Jared Strydhorst and Irène Buvat

IMIV, U1023 Inserm/CEA/Université Paris-Sud and ERL 9218 CNRS, Université Paris-Saclay, CEA/SHFJ, Orsay, France

E-mail: jared.strydhorst@gmail.com

Received 1 April 2016, revised 10 August 2016
Accepted for publication 12 August 2016
Published 2 September 2016

Abstract

The GATE software platform, based on the Geant4 toolkit for simulating particle interactions with matter, enables simulation of, among other medical imaging and treatment systems, positron emission tomography. However, at least one publication (Moraes *et al* 2015 *Phys. Med.* **31** 43–8) has reported discrepancies between the expected results and those obtained using GATE simulations, specifically with respect to the coincidence sorter which processes single events detected by the scanner to find coincidence pairs. In particular, the current software appears to overestimate the number of 'true' coincidence pairs when in multi-window mode, while the delayed coincidence window, used to estimate the randoms present in the prompt coincidence window, underestimates the randoms. Both effects are particularly evident at high count rates. We have investigated this discrepancy and reproduced the reported problems. We have also rewritten the relevant portion of the GATE code to correct the issue. In this note we describe the modifications to the coincidence sorter and repeat the simulations which previously showed unexpected results. Some discrepancies remain in the estimation of the randoms with the single-window mode which are a consequence of the algorithm itself. In multi-window mode however, the simulation agrees exactly with the expected results. The modifications to the coincidence sorter code will be incorporated into the next release of GATE (> version 7.2).

Keywords: GATE, PET, Monte Carlo, coincidence sorter, simulation

(Some figures may appear in colour only in the online journal)

Introduction

The GATE simulation toolkit (Jan *et al* 2004, 2011) is used extensively to model PET scanners and simulate acquisitions for research purposes. Essential to the simulation of clinical systems is the realistic simulation of the coincidence sorter, which processes the stream of single photons detected by the system to find pairs of singles detected within a set time interval, known as the prompt coincidence window.

Consider, for example, the situation shown in figure 1. If we consider the first single, S_1, we notice that both S_2 and S_3 fall within the specified prompt coincidence window. S_2 and S_3 are potentially in coincidence with S_4, which itself is in coincidence with S_5. One of two methods is typically used to define the coincidence window. In the case of the single-window (SW) method, once an incoming pulse has 'opened' a coincidence window no other pulse can open another until it has closed. In this case, S_1 will be detected as being in coincidence with S_2 and S_3. Once this window has closed, S_4 is able to open its own coincidence window and will be detected as being in coincidence with S_5. However, this method might miss legitimate coincidences. For example, in this scenario, singles S_3 and S_4 will not be detected as being in coincidence, even though they may have originated from a single positron annihilation.

Alternatively, the multiple-window (MW) approach permits each incoming pulse to open its own prompt coincidence window. In this case, S_1 will open a window and be detected to be in coincidence with S_2 and S_3, S_2 will open a window that will contain S_3 and S_4. S_3 will be detected as being in coincidence with S_4 and S_4 with S_5.

A policy for handling coincidence windows during which more than two singles are detected must also be chosen. Sometimes pairs of singles can be discarded for not forming valid coincidences, for example, if the singles forming the pair of events are detected too close together in the ring of detectors and would therefore correspond to a line of response (LOR) outside the FOV. In GATE, pairs of singles are rejected where the distance between the detectors falls below some specified minimum. At other times, three or more LORs joining the singles may potentially correspond to lines along which the annihilation actually occurred. Here the GATE software offers several options: for example, 'takeAllGoods', where every pair corresponding to the legitimate LOR is accepted as a valid coincidence event, 'takeWinnerOfGoods', where only the pair with the highest total energy deposition is recorded, 'killAllMultiples', which simply discards all multiple events, 'keepIfAllAreGood', where the event is passed on as a 'multiple' event that affects the scanner dead time, though the prompts themselves are discarded and not included in the output coincidence pairs, as well as several others. Processing the singles must be handled carefully to avoid double counting. In SW mode and 'takeAllGoods' option, using the above example, the software might create three prompts (S_1S_2, S_1S_3, and S_2S_3) from the first multiple (S_1, S_2, S_3). In MW mode, only pairs with the initial event S_1 (i.e. S_1S_2 and S_1S_3) are created for each window to avoid double counting, since the S_2S_3 event will show up in the coincidence window opened by S_2.

The processing also has to handle delayed windows in such a way that the detected counts in the delayed coincidence window provide an accurate estimate of the randoms detected in the prompt coincidence window.

Another challenge, specific to GATE simulations, is the problem of dealing with singles that do not necessarily arrive at the coincidence sorter in strict chronological order. This is irrelevant to a real scanner, but GATE generates events sequentially, following the daughter particles until they are detected, leave the simulation volume, or the energy falls below a certain threshold. Once an event has been completely processed, the next event is generated. It is possible for the second event to result in a single that is detected chronologically prior (in

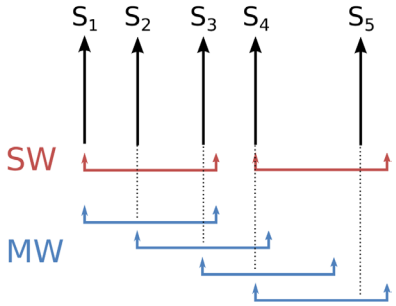

Figure 1. Single window (SW) and multiple window (MW) coincidence detection.

'simulation time') to one from a previously simulated event if, for example, it originated near a detector while the previous event was more distant and required more time to propagate. Moreover, GATE also allows the user to apply time resolution blurring to the detected singles which modifies the time stamps to simulate the inherent uncertainty in the temporal precision of the detector electronics. Therefore singles are not necessarily sorted in the chronological order they would have reached the coincidence sorter in a real experiment.

A couple of papers have been published on the validation of the coincidence sorter, suggesting some shortcomings that may be related to our observations of small numbers of missed coincidences and invalid events in the coincidence data. The first by Guez *et al* (2008), although it reported good agreement between simulation and experimental data for the total combined number of prompts and delayed events, also noted some divergence between the experimental and simulated prompt and delayed event rates with high activity levels. The second paper, Moraes *et al* (2015), reported some unexplained disagreements between the count rate of coincidences reported by GATE and those calculated offline, particularly for simulations of high activities.

Our own analysis also found similar inconsistencies in the existing coincidence sorter, which motivated the present work, in which we have completely rewritten the coincidence sorter. We repeated the experiments of Moraes *et al* (2015) using both the coincidence sorter in GATE 7.2 and our new coincidence sorter. The new version of the coincidence sorter corrects the observed deficiencies.

Methods

Coincidence sorter algorithm

The rewritten coincidence sorter uses two data structures. The first describes a single and contains, among other data, an event ID that indicates the annihilation (or other) event that created the single, the spatial coordinates of the single (where it was detected), and a time stamp indicating the time of detection. The other data structure is a coincidence window, which contains the start and end times for the window, and one or more single events. When coincidence windows are created, they initially contain the single event that 'opened' that window. The start and end time are set based on the time stamp of that initial single, the delay (0 in the case of the prompt coincidence window; positive and non-zero for a delayed coincidence window), and the window width. After a coincidence window is created, more single events can be added to it if they have a timestamp that falls between its start and end times.

The new coincidence sorter contains two buffers: a presort buffer that contains singles that have not yet been checked for coincidence with the already open coincidence windows, and a coincidence window buffer, containing the list of currently open coincidence windows.

The algorithm proceeds as follows:

1. Incoming singles are placed in the presort buffer. This buffer allows singles to be inserted in chronological order, with the earliest singles at the back end of the buffer, and the most recent at the front.
2. Once the buffer exceeds the specified size (default: 256, but can be modified by the user using the setPresortBufferSize command), the earliest singles are 'popped' from the back of the buffer and tested for coincidence with each of the open coincidence windows:
   a. If the single occurs after the end of a coincidence window, that window is removed from the coincidence window buffer and processed (see below). The single is then checked for coincidence against the next coincidence window in the list.
   b. If the single is in coincidence with a coincidence window (i.e. the time of the single falls between the start and end time of the coincidence window), a copy of the single is added to that window. A flag is set to indicate the single is in coincidence, and the single is checked against the next coincidence window.
   c. If the single is before the start time of a coincidence window, or if the end of the list of open coincidence windows has been reached, then:
      i. If in SW mode and the single was already in coincidence (the flag was set in b.), the single is discarded (the copy that was added to the window where it was in coincidence still exists).
      ii. Otherwise, the single is used to create a new coincidence window which is added to the list of open coincidence windows[1].

[1] Consider what happens in the case of the events shown in figure 1, operating in MW mode.

1. S_1 is popped from the presort buffer. Since there are no open coincidence windows (case c.), it is used to create a new coincidence window which we will label C_1, containing one event, S_1.
2. Next S_2 is tested and found to be in coincidence with the window C_1. A copy of S_2 is added to C_1 which now contains S_1 and S_2. Since the end of the list of coincidence windows has been reached, and the sorter is in MW mode, S_2 is used to create a new coincidence window, C_2.
3. S_3 is tested for coincidence with C_1 and C_2 and a copy is added to each one. When the end of the list is reached, S_3 is used to create coincidence window C_3.
4. S_4 is tested for coincidence with window C_1. Since it is after the end of C_1, C_1 (containing S_1 and copies of S_2 and S_3) is processed and removed from the list of open coincidence windows. S_4 is tested against C_2 and then C_3 and copies of S_4 are added to both of them, since they are in coincidence. S_4 is used to create a new coincidence window, C_4.
5. S_5 is tested for coincidence with C_2. The time of S_5 is after the end of C_2, so C_2 is processed and removed from the list. Likewise for C_3. S_5 is in coincidence with C_4, so a copy is added to C_4. A new window C_5 is created.
6. Suppose a single, S_6, not shown, arrives much later so it is not in coincidence with either C_4 or C_5. Windows C_4 (including S_4 and a copy of S_5) and C_5 (containing only a single, S_5) will be processed and removed from the list of windows. Window C_6 will be created, and left as the only window in the list of coincidence windows.

If the sorter is in SW mode, S_2 and S_3 would be discarded, rather than used to create C_2 and C_3 respectively.

In the case of a prompt coincidence window, the start time of the window is always the same as the time of the single used to create it.

For a delayed coincidence window, the window start time is always later than that of the initial single. The same algorithm is used to check for coincidence. For example, S_1 would potentially create a window C_1 with a start time that for the sake of this example falls between S_4 and S_5. When event S_2 is tested for coincidence, it is found to occur before the start time of C_1, so a new coincidence window is opened and added to the list, again with a delayed start time, so C_2 is after C_1. Likewise for S_3 and S_4. At this point the list of coincidence windows contains C_1, C_2, C_3, and C_4. When S_5 is checked, it will be found to be in coincidence with C_1, so a copy is added to C_1, etc.

Table 1. Parameters of the GATE model of the Siemens Biograph scanner.

Crystal material	LSO
Crystal size	$4 \times 4 \times 20$ mm
Ring diameter	842 mm
Crystals/ring	624
Rings	52
Energy resolution	11.6% @ 511 keV
Quantum efficiency	0.8
Pileup time	120 ns
Dead time	640 ns
Time resolution blurring	0.5275 ns
Coincidence window width	4.1 ns
Delayed window shift	500 ns
Energy window	435–650 keV

Processing of coincidence windows involves discarding those which contain only the original single and no others, verifying that those which contain two singles—the original plus one additional—are valid prompts (i.e. they correspond to a LOR within the FOV), and processing those that contain more than two singles according to the specified multiples policy.

Since there is no mechanism to flush the presort buffer at the end of the simulation, this approach always discards a number of singles at the end of each simulation equal to the size of the presort buffer. For realistic simulations, this is unlikely to have any measurable influence on the results.

Simulations

A GATE model of the Biograph mCT scanner was created with the parameters shown in table 1. The intrinsic LSO activity was ignored since it is irrelevant for the purposes of this study.

The phantom simulated was the NEMA NU-2 2007 scatter phantom, comprising a polyethylene cylinder of 200 mm diameter by 700 mm long centered in the scanner FOV. A cylindrical line source, 3.2 mm diameter by 700 mm long, was simulated 45 mm off center. The ^{18}F source was simulated as a positron source with a kinetic energy distribution matching that of ^{18}F (the energy distribution in the GATE script is specified as 'Fluor18'). Simulations were run for 11 different source activities ranging from 10 MBq to 1 GBq. The simulation time was chosen for each simulation to obtain at least 3 million prompt counts. The simulation was run in SW and MW modes using both GATE version 7.2 and our own version with the modified coincidence sorter. A multiples policy of 'takeAllGoods' was used for all simulations.

The output of the simulation was stored in the ROOT file format (Brun and Rademakers 1997), developed by CERN for storing and analyzing large data sets typically consisting of lists of particle interactions simulated or detected. Each event detected in the simulation is stored with, among other data, a unique ID indicating the original event (eventID), the spatial coordinates of the original event and the detected photon, the time of the detected event (time), and whether the photon was scattered before detection. All analysis was done using ROOT 5.34.10.

Analysis

We analyze first the distribution of time differences between the singles making up the random events detected in the prompt coincidence window.

Random coincidences in the prompts coincidence window are determined by counting those where the two singles have different eventIDs. These are later compared to the estimate of the randoms rate obtained from the delayed coincidence window.

The rate of true events was determined by processing the coincidence pairs and extracting those where both singles originated from the same event (identical eventIDs). In this work, we did not check whether the photons contributing to coincidences have been scattered or not, thus, for the purposes of this paper, 'trues' also includes scatter. This is compared with the measured 'trues' coincidence rates derived by subtracting the delayed count rate from the prompts count rate.

Results

Figure 2 shows a histogram of the time differences for the random events detected in the prompt coincidence window, i.e. (time 2–time 1), for the 1 GBq simulation.

Figure 3 shows the trues coincidence rates detected by the various coincidence sorters, defined as those events in the prompt coincidence window where the eventID tags for both photons match, indicating that the photons originate from a single positron annihilation.

The rate of randoms in the prompt coincidence window (events where the eventIDs of the two photons do not agree) and the rate of events detected in the delayed coincidence window are shown in figure 4.

Figure 5 shows the count rate of the 'trues' (as in figure 3) compared with the rate of trues obtained by subtracting the events in the delayed coincidence window from the prompt coincidence window. The agreement of the new coincidence sorter is markedly improved in SW mode and almost perfect in MW mode.

Discussion

Figure 2 alone demonstrates a significant problem in the existing coincidence sorter. Theoretically, the time difference for the random events in the prompt coincidence window should be uniformly distributed over the duration of the window, in this case 4.1 ns. While the new coincidence sorter produces a uniform distribution, the existing coincidence sorter produces a non-uniform distribution of events and a significant number of events with time differences that fall outside of the 4.1 ns window.

Measuring the number of true events detected in the prompt window with the existing coincidence sorter, we have reproduced almost exactly the results of Moraes *et al* (2015) (see figure 3 of this work and figure 4 of Moraes *et al*). In particular, we observed that the existing coincidence sorter detects a large excess of true events in the prompt window in MW mode. With our modified coincidence sorter, the number of true events increases only slightly when using MW mode relative to SW mode. This is consistent with the expected behavior, since the number of trues should be the same in both cases, though in SW mode a very small number of trues might be missed when they occur when a window has already been opened. Referring to figure 1, consider the case where S_3–S_4 corresponds to a true event, which would be missed in SW mode, but detected in MW mode. This effect should be very small for the count rates simulated, on the order of no more than 1–2% for the situation

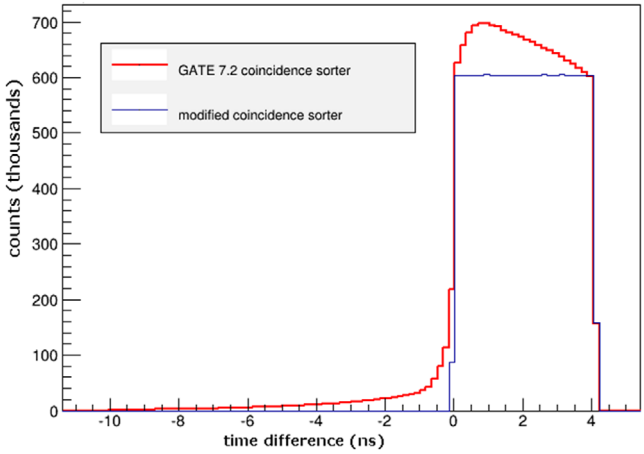

Figure 2. Histogram of time differences for the random events detected in the prompts coincidence window (coincidence events where the eventID's of the two single events are different) for the simulation with 1 GBq of activity, MW mode.

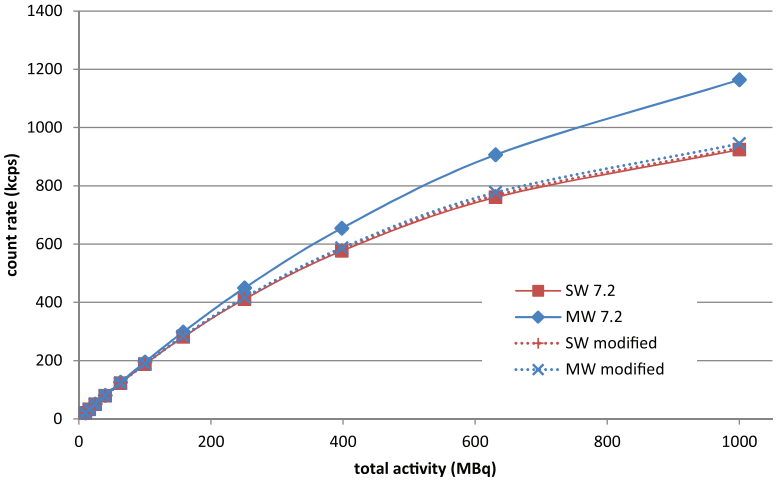

Figure 3. The count rate of 'true' events detected in the prompts coincidence window; coincidences where both photons originated from a single positron annihilation. Solid lines correspond to the current (GATE 7.2) coincidence sorter; dashed lines correspond to the modified coincidence sorter. Red lines are used for SW mode, blue lines for MW mode.

simulated[2]. With the old coincidence sorter, the difference was as high as 26%. With the new it is at most 1.5%. Moreover, Moraes *et al* reported that the rate of trues detected in SW mode agreed well with the rate calculated by offline processing of the singles. By implication, the new coincidence sorter produces a much more accurate measure of the

[2] This can happen when one single of a true pair arrives near the end of an already open window and the correlated single, although detected, arrives after the end of the window. Thus, the probability is on the order of magnitude of the singles rate multiplied by the time difference between the two singles making up a true event. The latter follows a distribution that will depend on the physical distribution of activity and on the electronics of the detectors, so a rigorous calculation would be a challenge, but, for example, with a singles rate of 10^7 s^{-1} and an average temporal separation between a true pair of 1 ns, the loss would be approximately 1%.

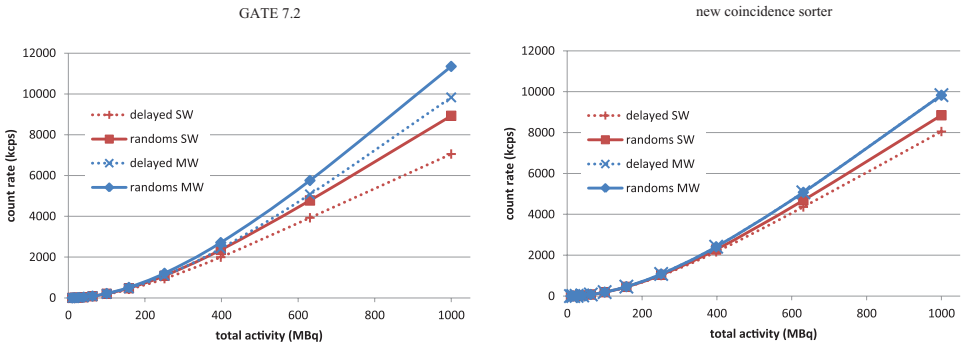

Figure 4. The actual rate of randoms in the prompt coincidence window (solid lines) compared with the rate of randoms in the delayed coincidence window (dashed lines). Left: GATE 7.2 coincidence sorter, right: new coincidence sorter. Red lines are used for SW mode, blue lines for MW mode.

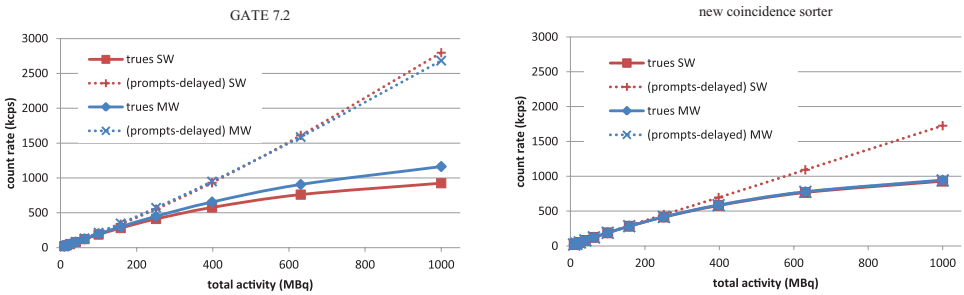

Figure 5. The actual number of trues (solid lines) and the number of 'trues' calculated from the difference between the number of events detected in the prompt and the delayed coincidence windows (dashed lines). Left: GATE 7.2 coincidence sorter, right: new coincidence sorter. Red lines are used for SW mode, blue lines for MW mode.

true events in MW mode than the existing one. The results are also quantitatively similar to experimental data for the Biograph mCT from Rausch *et al* (2015) where for the same setup with approximately 1 GBq of activity, the combined trues and randoms count rate was approximately 1000 kbps (see figure 3(a) of that paper). However given the uncertainty of the modelling of the dead-time, and the omission of the LSO background from our simulations, rigorous quantitative comparison is not possible.

With the old coincidence sorter, the actual number of randoms in the prompt coincidence window is significantly underestimated by the number of events detected in the delayed coincidence window (figure 4). With the new coincidence sorter the randoms are still slightly underestimated in SW mode, though the agreement is significantly better than before. Using the new coincidence sorter in MW mode the number of events in the delayed coincidence window agrees almost perfectly with the number of randoms in the prompt coincidence window.

As figure 5 shows, with high levels of activity, there is a substantial error in the trues rate when estimated from the difference between the prompts and the delayed events. With the new coincidence sorter, the difference is effectively eliminated in MW mode. In SW mode, the difference is much less than previously, though still present. The reason is an underestimate of the randoms rate by the delayed window at high activity levels. In practice, most modern PET scanners employ the multi-window mode (Moraes *et al* 2015).

In single window mode with the 'takeAllGoods' policy, the delayed coincidence window alone cannot correctly estimate the randoms for high activity levels. There are two options for processing the counts recorded in the delayed coincidence window. Option 1 is to treat the delayed coincidence window exactly as the prompt coincidence window, where all valid pairs of singles are considered as delayed events, including those pairs consisting of two events in the delayed window. Alternatively, in option 2, only those pairs of events where there is a delay between the two single events, i.e. pairs where one of the events is the initial event which 'opened' the delayed coincidence window, can be considered valid.

If all pairs are allowed (option 1), a delayed coincidence window containing two events in the window itself, plus the initial event will produce three coincidence pairs. This correctly estimates the number of randoms that arise from the situation where the prompt coincidence window contains three uncorrelated events, but a true coincidence in the delayed coincidence window will also contribute to the randoms estimate, overestimating the randoms.

If, on the other hand, the events in the delayed coincidence window are only paired with the original event and not with each other (option 2), the inclusion of trues in the randoms estimate is avoided, but this also under-corrects for the situation where all three events are uncorrelated, since that situation produces three coincidences in the prompt window, but only two in the delayed window. The discrepancy is even more significant for higher order multiples, though in practice these are extremely rare even at relatively high activity levels.

For relatively low counts rates, the discrepancy between the randoms estimated from the delayed window and the true number of randoms in the prompts coincidence window is low, but as our simulations show, the divergence becomes much larger as the count rate increases. For better estimation of randoms, either SW mode should be used with a different multiples policy such as 'takeWinnerOfGoods' or 'killAllMultiples'. The 'takeAllGoods' policy should in principle only be used in MW window mode.

The range of activities tested in the work are admittedly beyond the range of clinically plausible scans. Below 100 MBq, the discrepancies due to inaccuracies in the coincidence sorter are relatively smaller, though still apparent from careful observation of the data. However for simulation of situations with a high random fraction where very accurate correction is desirable, ^{90}Y PET for example, accurate modelling of the coincidence sorter is essential to obtain reliable simulation results.

Though we have not recreated the simulations of Guez *et al* (2008), our results would likely explain, at least in part, the discrepancies they demonstrate between the simulated and experimental data at high activity, where they show a clear overestimation of prompts by the simulation and an underestimate of delayed counts.

Conclusion

We have rewritten the GATE coincidence sorter. The new coincidence sorter offers much better estimates of the number of randoms and trues in the prompts coincidence window which now agree with off-line processing of the single events, particularly in multi-window mode, which previously exhibited significant disagreement. The estimate of the randoms provided by the delayed coincidence window now also agrees much better with the number of random events in the prompts coincidence window in multi-window mode with a multiples policy of 'takeAllGoods'. Single window mode with the 'takeAllGoods' policy is not recommended.

Acknowledgments

J Strydhorst was funded by the INCA PhysiCancer 2014 MILADY contract.

References

Brun R and Rademakers F 1997 ROOT—an object oriented data analysis framework *Nucl. Instrum. Methods Phys. Res.* A **389** 81–6

Guez D, Bataille F, Comtat C, Honoré P F, Jan S and Kerhoas S 2008 Counting rates modeling for PET scanners with GATE *IEEE Trans. Nucl. Sci.* **55** 516–23

Jan S *et al* 2004 GATE: a simulation toolkit for PET and SPECT *Phys. Med. Biol.* **49** 4543–61

Jan S *et al* 2011 GATE V6: a major enhancement of the GATE simulation platform enabling modeling of CT and radiotherapy *Phys. Med. Biol.* **56** 881–901

Moraes E R, Poon J K, Balakrishnan K, Wang W and Badawi R D 2015 Towards component-based validation of GATE: aspects of the coincidence processor *Phys. Med.* **31** 43–8

Rausch I, Cal-González J, Dapra D, Gallowitsch H J, Lind P, Beyer T and Minear G 2015 Performance evaluation of the Biograph mCT Flow PET/CT system according to the NEMA NU2-2012 standard *EJNMMI Phys.* **2** 26