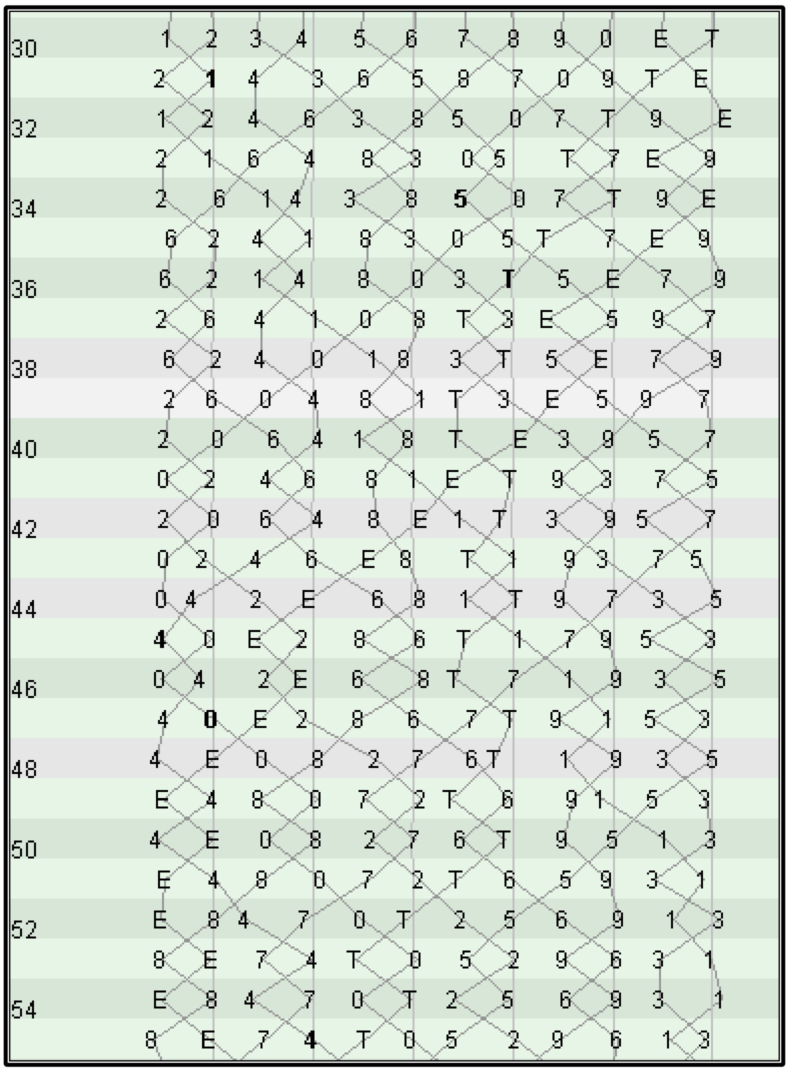
The Liverpool Ringing Simulator

A Virtual Striking Competition



Author: Andrew Instone-Cowie

Date: 04 January 2020
Version: 0.1

Contents

Index of Figures 3
Document History 4
Licence 4
Background 5
  Polling Design 5
  The Variable Odd-Struckness Problem 5
  Virtual Striking Competition 6
Virtual Competition Scenarios 7
  Test Scenario A 7
  Test Scenario B 8
  Test Scenario C 9
  Interim Results 10
Real World Striking Comparison 11
  Scenario F – Crediton 2010 First Placed Band 11
  Scenario G – Crediton 2010 Fifth Placed Band 11
  Scenario H – Crediton 2010 Ninth Placed Band 12
  Overall Result 12
Acknowledgements 13

Index of Figures

Figure 1 – Variable Odd-Struckness Illustration 5
Figure 2 – Test Scenario A 7
Figure 3 – Scenario A Test Results 7
Figure 4 – Test Scenario B 8
Figure 5 – Scenario B Test Results 8
Figure 6 – Test Scenario C 9
Figure 7 – Scenario C Test Results 9
Figure 8 – Interim CAS Results 10
Figure 9 – Crediton 1st Placed Band 11
Figure 10 – Crediton 5th Placed Band 11
Figure 11 – Crediton 9th Placed Band 12
Figure 12 – Overall CAS Results 12

Document History

Version	Author	Date	Changes
0.1	A J Instone-Cowie	04/01/2020	First Draft.

Licence

[1] http://creativecommons.org/licenses/by-sa/4.0/

Background

Polling Design

Both the Type 1 and Type 2 variants of the Liverpool Ringing Simulator use a polling approach to detecting sensor inputs. This is discussed in detail in the Type 2 ***Technical Reference Guide***, but in summary, after completing initialization, the firmware code cycles round all the active inputs, examining the state of each sensor in turn. After all inputs have been read, the code loops round and the process starts again.

In the Type 2 Simulator Interface Module each iteration of the main polling loop takes approximately 400µs when configured to poll all 16 possible sensors. Put another way, the inputs are polled at approximately 2.5kHz.

The Variable Odd-Struckness Problem

This use of a polling architecture invariably introduces a degree of variable odd-struckness.

- If a pulse from any Sensor Module starts a fraction of a second before the polling loop examines the associated input, the pulse will be detected almost immediately.
- If the pulse starts a fraction of a second after the polling loop examines the associated input, the pulse will not be detected until the next iteration of the polling loop, potentially as much as 400µs later.
- There is no fixed correlation between the start time of a pulse and the current position of the polling loop in its cycle, and therefore each pulse is subject to an effectively random delay of between zero and 400µs (the duration of the polling loop).
- The mean delay would be 200µs (half the polling interval), so if this was detectable then it would result in each simulated bell apparently striking randomly early or late by between zero and 200µs.

The following diagram taken from Type 2 ***Technical Reference Guide*** illustrates this problem:

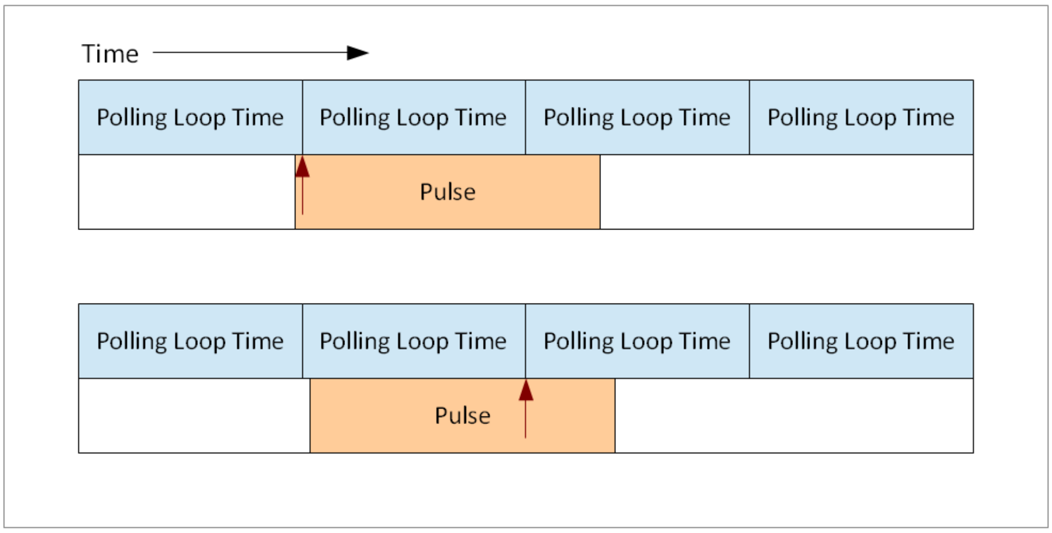


Figure 1 – Variable Odd-Struckness Illustration

The upper red arrow indicates the polling loop examining an input just after the start of an incoming sensor pulse. The delay due to polling is effectively zero.

In the lower diagram, the polling loop examines the input pin just before the start of an incoming sensor pulse, and hence does not detect the pulse until the next iteration of the loop, indicated by the lower red arrow. The delay due to polling is effectively equal to the polling loop interval.

The design of the simulator assumes that this variable odd-struckness is too small to be significant and is in practice not detectable.

The first question this paper seeks to consider is: ***Is this assumption reasonable?***

Virtual Striking Competition

The usual ringing approach to quantifying errors in striking is to conduct a striking competition.

A "virtual striking competition" was conducted to validate the reasonableness of this assumption. This exercise pre-dated the development of the Type 2 Simulator, so was undertaken with a Type 1 Simulator Interface driving Abel[2] running on a PC, however both hardware variants use the same underlying approach.

The competition was conducted as follows:

- The "test piece" consisted of 10 minutes of rounds on 12, with an inter-bell interval of approximately 200ms and an open handstroke lead of 1.0. This results in approximately 240 rows at a peal speed of 3h 30m.
- The Abel striking statistics were reset prior to each test, the striking for each test piece was recorded, and then exported from Abel in "*Lowndes*" text format[3].
- Each Lowndes file was then imported into the CAS[4] tool for analysis. CAS (*Computer Analysis of Striking*) is used in conjunction with the *Hawkear*[5] system as part of the judging process for the *National 12-Bell Striking Competition*[6].
- The individual results for each test piece were recorded, and all test pieces were then ranked using the CAS Band Summary tool.
- Three different scenarios were tested, and these are detailed below.
- Note that data relating to two further scenarios has been removed for clarity; this related to other experimental hardware and is not relevant to this discussion.

[2] http://www.abelsim.co.uk/
[3] http://www.abelsim.co.uk/doc/striking.htm
[4] http://www.12bell.org.uk/downloads/cas1.4.zip, released under GNU General Public Licence v3.
[5] http://www.12bell.org.uk/hawkear/
[6] http://www.12bell.org.uk/

Virtual Competition Scenarios

Test Scenario A

Test Scenario A was a control scenario, using Abel alone with no simulator hardware.

- Abel was setup to ring rounds on 12 at the required speed, the statistics cleared, and the ringing started.
- No external simulator hardware was used in this test.
- In this test, as in all tests, the Simulator PC was left as undisturbed as possible during ringing.

Scenario A is illustrated in the following diagram:

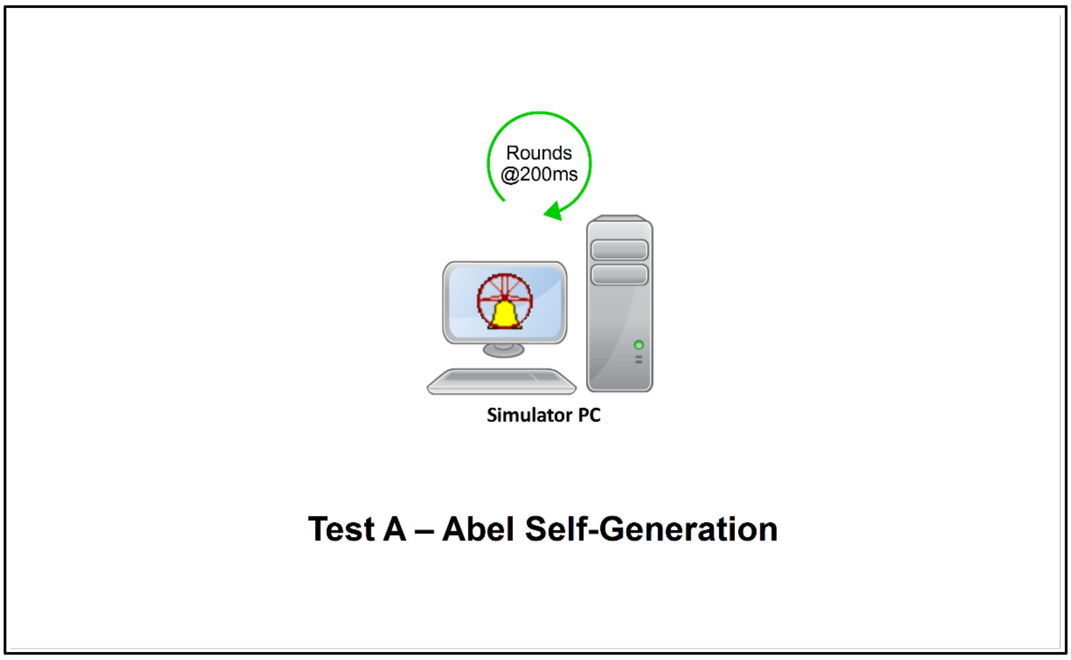
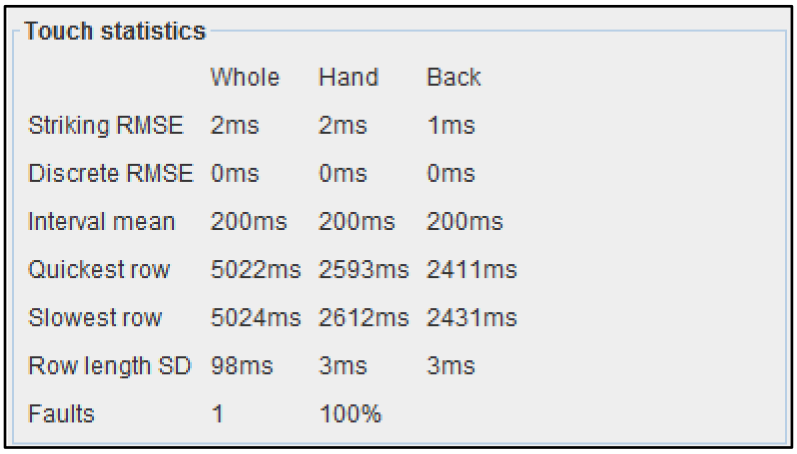


Figure 2 – Test Scenario A

The CAS analysis of the striking from Test Scenario A is shown in the following table. The single fault is most likely the result of some background processing on the PC.



Touch statistics

	Whole	Hand	Back
Striking RMSE	2ms	2ms	1ms
Discrete RMSE	0ms	0ms	0ms
Interval mean	200ms	200ms	200ms
Quickest row	5022ms	2593ms	2411ms
Slowest row	5024ms	2612ms	2431ms
Row length SD	98ms	3ms	3ms
Faults	1	100%	

Figure 3 – Scenario A Test Results

Test Scenario B

Test Scenario B was also a control and tested the accuracy of serial input detection in Abel.

- Abel was driven by a Simulator Interface operating with custom test code.
- The Interface was setup to ring rounds on 12 at the required speed, all non-essential code being eliminated. The Abel statistics were cleared, and the ringing started.
- No simulator sensor inputs were read in this test, and hence this test should not be susceptible to the variable odd-struckness problem.

Scenario B is illustrated in the following diagram:

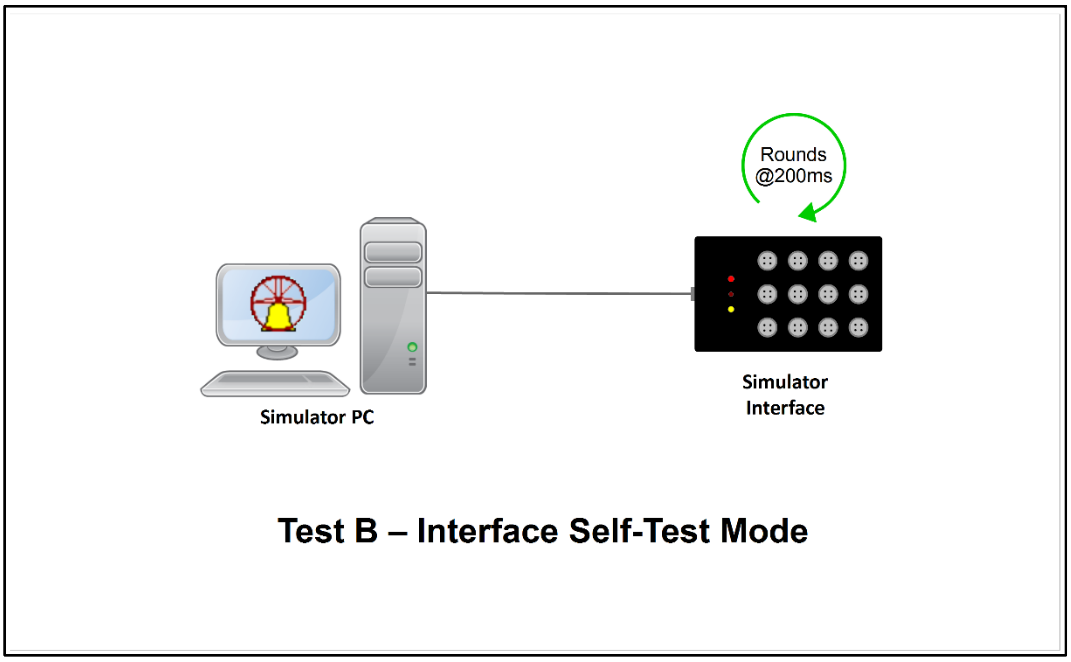
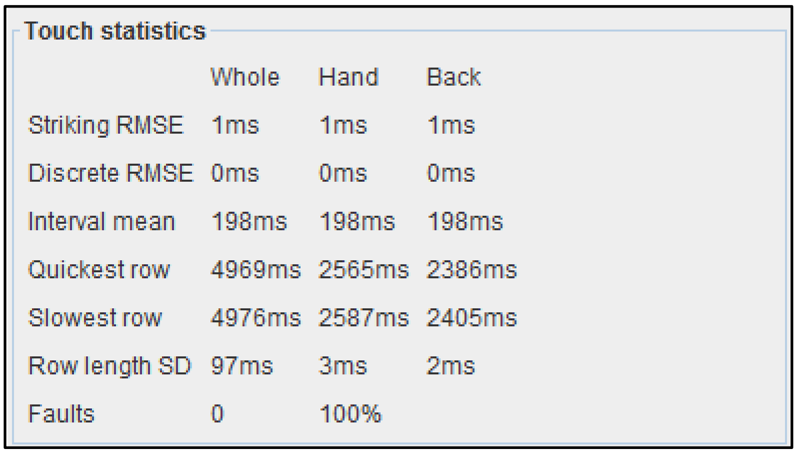


Figure 4 – Test Scenario B

The CAS analysis of the striking from Test Scenario B is shown in the following table.



Touch statistics

	Whole	Hand	Back
Striking RMSE	1ms	1ms	1ms
Discrete RMSE	0ms	0ms	0ms
Interval mean	198ms	198ms	198ms
Quickest row	4969ms	2565ms	2386ms
Slowest row	4976ms	2587ms	2405ms
Row length SD	97ms	3ms	2ms
Faults	0	100%	

Figure 5 – Scenario B Test Results

Test Scenario C

Test Scenario C introduced susceptibility to the variable odd-struckness problem.

- Abel was driven by a Simulator Interface operating with standard code. The Interface sensor inputs were driven by an Arduino setup as a pulse generator, to ring rounds on 12 at the required speed. The Abel statistics were cleared, and the pulse generator started.
- 12 simulator sensor inputs were polled in this test, and hence the test should be susceptible to the variable odd-struckness problem.
- This was the most live-like of the main test scenarios, including all interface code and polling variability.

Scenario C is illustrated in the following diagram:

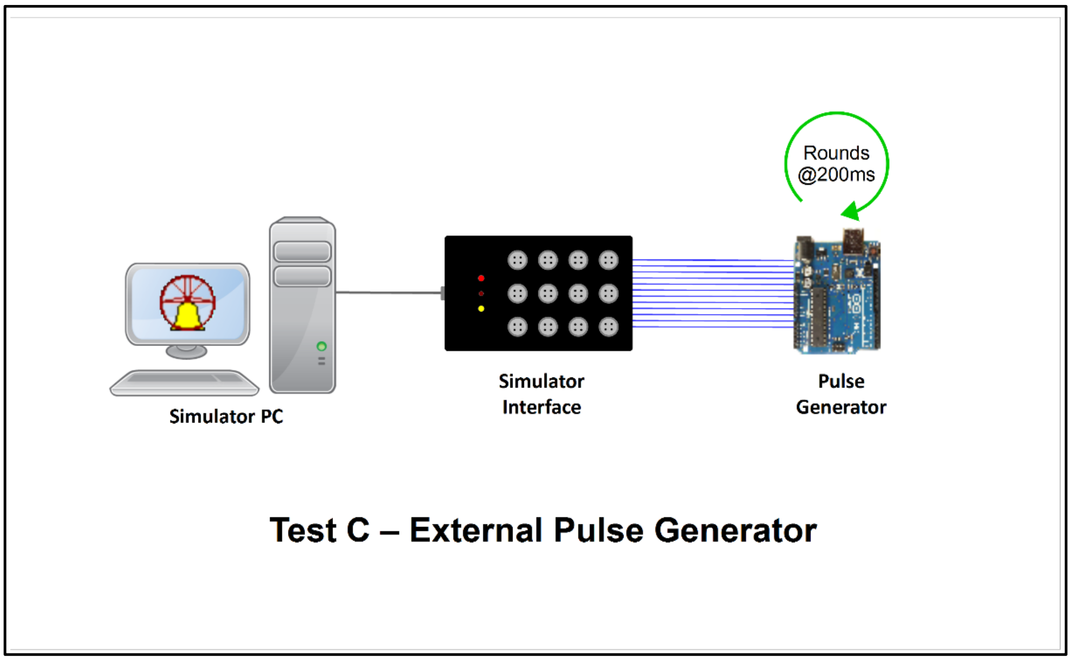
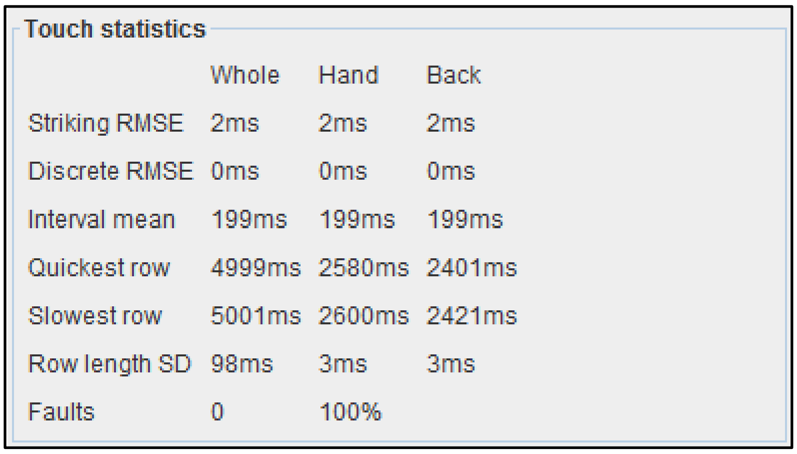


Figure 6 – Test Scenario C

The CAS analysis of the striking from Test Scenario C is shown in the following table. The virtual ringing is still very good, and no faults are recorded.



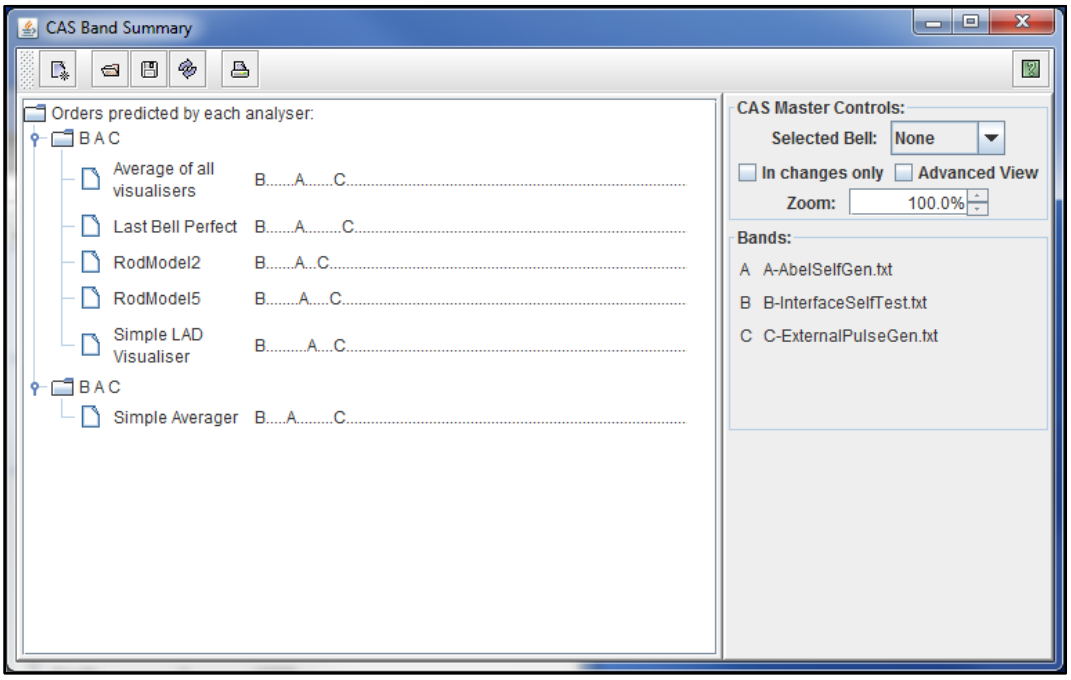
Touch statistics

	Whole	Hand	Back
Striking RMSE	2ms	2ms	2ms
Discrete RMSE	0ms	0ms	0ms
Interval mean	199ms	199ms	199ms
Quickest row	4999ms	2580ms	2401ms
Slowest row	5001ms	2600ms	2421ms
Row length SD	98ms	3ms	3ms
Faults	0	100%	

Figure 7 – Scenario C Test Results

Interim Results

The CAS Band Summary results for all three scenarios are shown in the following table:



```
CAS Band Summary

Orders predicted by each analyser:
  B A C
    Average of all visualisers   B......A......C......................
    Last Bell Perfect            B......A........C....................
    RodModel2                    B......A..C..........................
    RodModel5                    B.......A....C.......................
    Simple LAD Visualiser        B.........A....C.....................
  B A C
    Simple Averager              B....A........C......................

CAS Master Controls:
  Selected Bell: None
  [ ] In changes only   [ ] Advanced View
  Zoom: 100.0%

Bands:
  A  A-AbelSelfGen.txt
  B  B-InterfaceSelfTest.txt
  C  C-ExternalPulseGen.txt
```

Figure 8 – Interim CAS Results

The detailed results above for each test scenario show that the striking achieved by the simulator is very good.

- Slightly surprisingly, the most accurate striking was achieved using the serial input (Scenario B) and not from Abel self-generating ringing (Scenario A). However, the difference between the two scenarios is extremely small.
- As expected, a very slight reduction in striking accuracy is introduced by polling Simulator Interface inputs (Scenario C versus Scenario B). The difference is again extremely small, and the striking still achieves a "perfect" 100% accuracy with no faults.
- Scenarios D and E have been removed, as noted above.

The assumption that any variable odd-struckness introduced by polling inputs is too small to be significant does therefore seem to be reasonable.

A more interesting question might be: ***How does the simulator striking compare to the best that humans can achieve?***

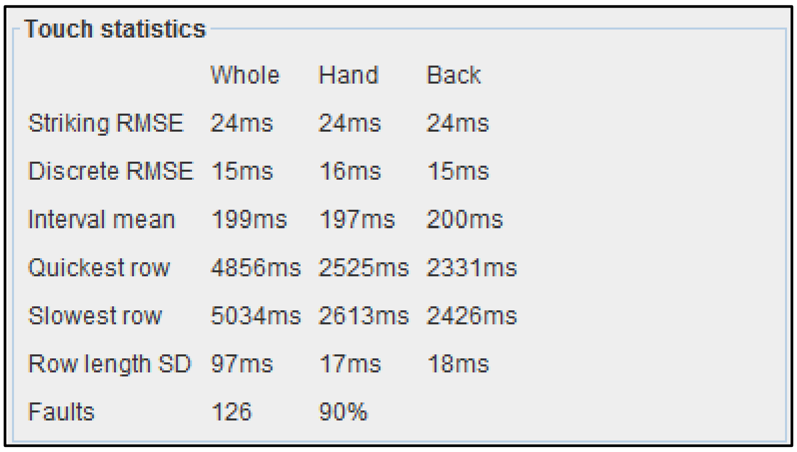
Real World Striking Comparison

To consider the second question, the results of the virtual striking competition were compared with the actual striking data from the bands placed first, fifth and ninth (i.e. last) in the 2010 *National 12-Bell Striking Competition* held at Crediton. This striking data is used for testing by the CAS developer[7].

This comparison does not pretend to be a detail statistical analysis and should be treated with a degree of caution, but it does provide a rough indication of the relative magnitudes of the errors introduced by the simulator, and of the errors naturally incurred by highly experienced real ringers ringing under competition conditions.

The speed of competition ringing and the overall number of changes rung are broadly comparable with the simulator test scenarios above[8].

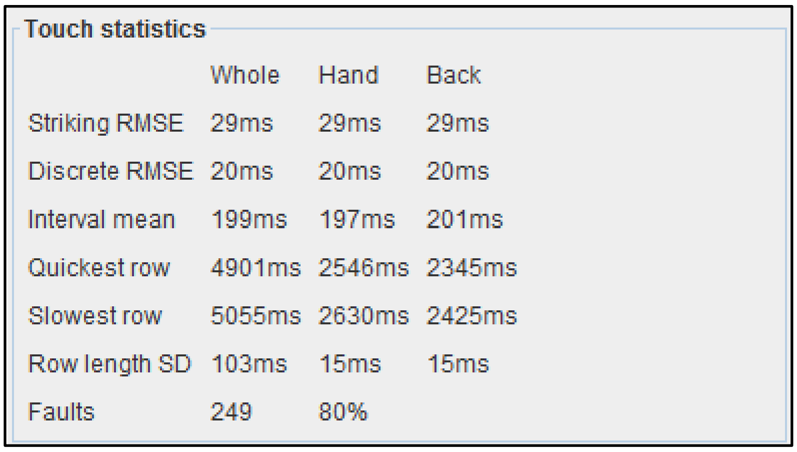
Scenario F – Crediton 2010 First Placed Band


Touch statistics

	Whole	Hand	Back
Striking RMSE	24ms	24ms	24ms
Discrete RMSE	15ms	16ms	15ms
Interval mean	199ms	197ms	200ms
Quickest row	4856ms	2525ms	2331ms
Slowest row	5034ms	2613ms	2426ms
Row length SD	97ms	17ms	18ms
Faults	126	90%	

Figure 9 – Crediton 1st Placed Band

Scenario G – Crediton 2010 Fifth Placed Band

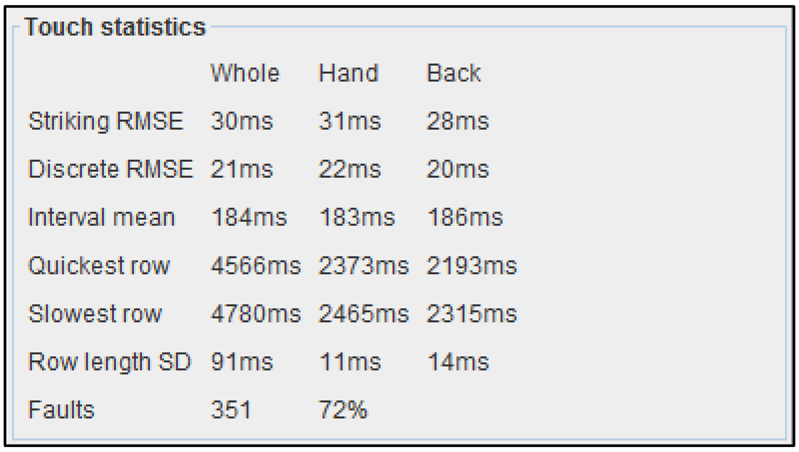

Touch statistics

	Whole	Hand	Back
Striking RMSE	29ms	29ms	29ms
Discrete RMSE	20ms	20ms	20ms
Interval mean	199ms	197ms	201ms
Quickest row	4901ms	2546ms	2345ms
Slowest row	5055ms	2630ms	2425ms
Row length SD	103ms	15ms	15ms
Faults	249	80%	

Figure 10 – Crediton 5th Placed Band

[7] https://github.com/EmBeeDee/CAS
[8] http://www.12bell.org.uk/cgi-bin/results.cgi?year=2010&venue=crediton. Note that the judges' final scores are not derived solely from CAS.

Scenario H – Crediton 2010 Ninth Placed Band



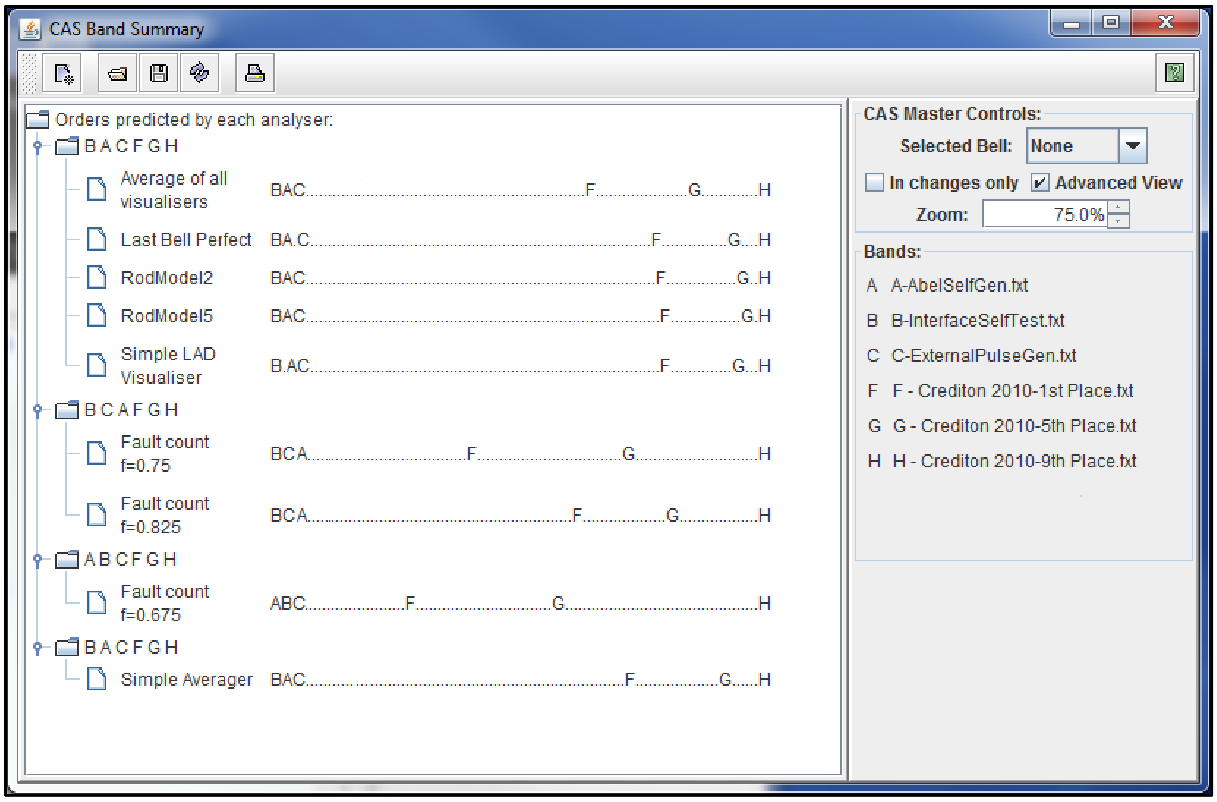
Touch statistics

	Whole	Hand	Back
Striking RMSE	30ms	31ms	28ms
Discrete RMSE	21ms	22ms	20ms
Interval mean	184ms	183ms	186ms
Quickest row	4566ms	2373ms	2193ms
Slowest row	4780ms	2465ms	2315ms
Row length SD	91ms	11ms	14ms
Faults	351	72%	

Figure 11 – Crediton 9th Placed Band

Overall Result

The CAS summary results for all virtual and competition scenarios are shown in the following table:



CAS Band Summary

```
Orders predicted by each analyser:
B A C F G H
    Average of all visualisers   BAC......................................F...............G.........H
    Last Bell Perfect            BA.C..............................................F..........G...H
    RodModel2                    BAC...............................................F...........G.H
    RodModel5                    BAC................................................F...........G.H
    Simple LAD Visualiser        B.AC...............................................F.........G...H
B C A F G H
    Fault count f=0.75           BCA.......................F.......................G....................H
    Fault count f=0.825          BCA......................................F..............G..............H
A B C F G H
    Fault count f=0.675          ABC..............F......................G..............................H
B A C F G H
    Simple Averager              BAC..............................................F..............G.....H
```

CAS Master Controls:

Selected Bell: None

In changes only ☐ Advanced View ✓

Zoom: 75.0%

Bands:

A	A-AbelSelfGen.txt
B	B-InterfaceSelfTest.txt
C	C-ExternalPulseGen.txt
F	F - Crediton 2010-1st Place.txt
G	G - Crediton 2010-5th Place.txt
H	H - Crediton 2010-9th Place.txt

Figure 12 – Overall CAS Results

As might be expected, all the simulator results are bunched up at the left side of the analysis, showing that the errors introduced by the simulator are very much smaller than the errors naturally incurred by the human ringers.

Overall, the assumption that variable odd-struckness introduced by polling simulator inputs is too small to be significant does appear to be justified.

Acknowledgements

CAS is developed and released under the GPL by Mark B Davies.

Abel is developed by Chris Hughes and Simon Feather, and is copyright © 2019 AbelSim Ltd.