CAS 741: Test Report

Aqueous Speciation Diagram Generator

Steven Palmer
palmes4

December 19, 2017

Revision History

Table 1: Revision History

Date	Developer(s)	Change
12.18.2017	S. Palmer	Revision 1

Contents

This document gives the results of the testing proposed in the SpecGen Test Plan document (found here).

1 Functional Requirements Evaluation

1.1 Automated Tests

T1: Diagram generation of $FeOH_3$ system: PASS
T2: Diagram generation of CO_2 system: PASS

1.2 Manual Tests

T3: Comparison of generated speciation diagram to original: PASS

Remarks: The output of the original MATLAB implementation is shown in Figure 2 and the output of SpecGen is shown in Figure 3 (see Appendix). A visual inspection of these diagrams reveals that they are the same. Note that the y-axis is different between these diagrams. This is not a problem since "fraction of total Fe" is simply the concentration divided by the total amount of Fe in the system. The total amount of Fe is a constant, and thus the shape of the curves is not affected.

2 Nonfunctional Requirements Evaluation

2.1 Manual Tests

T4: Readability of generated speciation diagram: PASS

Remarks: The SpecGen output for the $FeOH_3$ system is given in Figure 3 (see Appendix). Upon visual inspection, the title, axis labels, legend, and curves are all easily read, as required.

3 Unit Testing

T5: Plotting test: PASS
T6: Input conversion of $FeOH_3$ system: PASS
T7: Input conversion of CO_2 system: PASS
T8: Input conversion of empty system: PASS
T9: Calculation of empty system: PASS
T10: Calculation of simple system: PASS
T11: Register reaction as empty string: PASS
T12: Register reaction without equilibrium constant: PASS

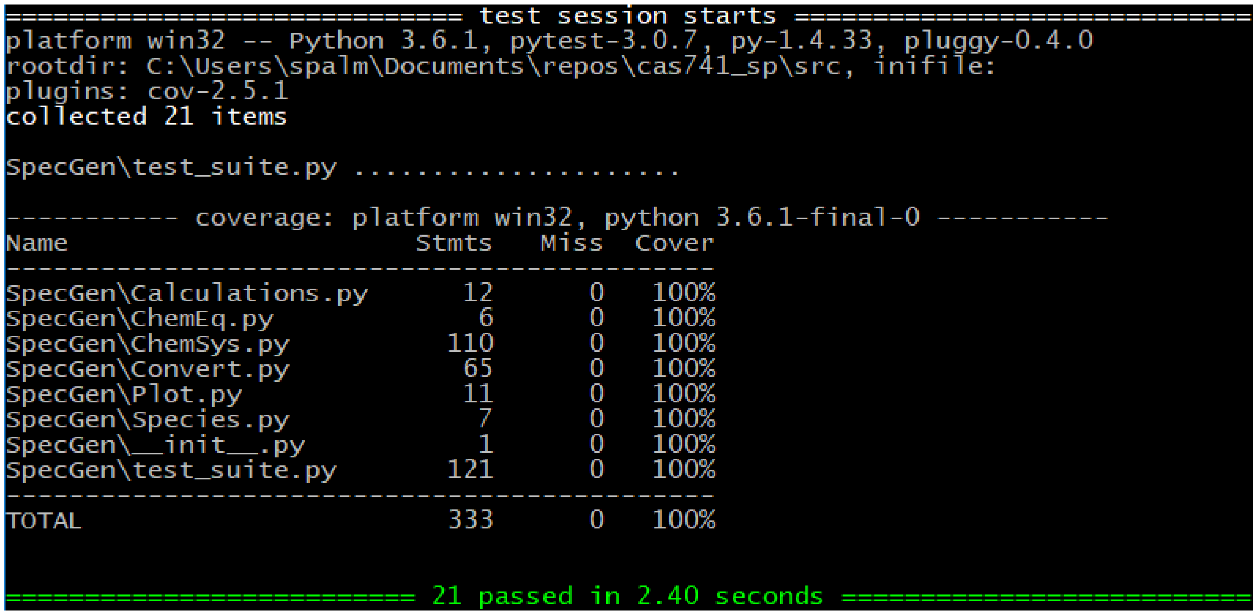
T13: Register reaction without products: PASS
T14: Register reaction with bad state: PASS
T15: Register reaction with bad formula (non-letter symbol): PASS
T16: Register reaction with bad formula (beginning with lower case): PASS
T17: Register reaction with bad formula (no parentheses): PASS
T18: Register reaction with bad formula (unbalanced parentheses): PASS
T19: Register reaction with superfluous parentheses: PASS
T20: Register reaction with high parenthesis nesting: PASS
T21: Register negative element total: PASS
T22: Register zero element total: PASS
T23: Register positive element total: PASS

4 Changes Due to Testing

Testing was carried out throughout development, with changes made to ensure SpecGen behaved as expected.

5 Automated Testing

The output of the automated test suite is shown in Figure 1. All automated tests passed.



```
============================= test session starts =============================
platform win32 -- Python 3.6.1, pytest-3.0.7, py-1.4.33, pluggy-0.4.0
rootdir: C:\Users\spalm\Documents\repos\cas741_sp\src, inifile:
plugins: cov-2.5.1
collected 21 items

SpecGen\test_suite.py .....................

----------- coverage: platform win32, python 3.6.1-final-0 -----------
Name                          Stmts   Miss  Cover
-------------------------------------------------
SpecGen\Calculations.py          12      0   100%
SpecGen\ChemEq.py                 6      0   100%
SpecGen\ChemSys.py              110      0   100%
SpecGen\Convert.py               65      0   100%
SpecGen\Plot.py                  11      0   100%
SpecGen\Species.py                7      0   100%
SpecGen\__init__.py               1      0   100%
SpecGen\test_suite.py           121      0   100%
-------------------------------------------------
TOTAL                           333      0   100%


========================== 21 passed in 2.40 seconds ==========================
```

Figure 1: Automated testing results

6 Trace to Requirements

A trace between system tests and requirements is provided in Table 2.

Table 2: Requirements Traceability

Requirement	**Test(s)**
R1	T1, T2, T3
R2	T1, T2, T3
R3	T1, T2, T3
R4	T1, T2, T3
R5	T1, T2, T3
NF1	T4

7 Trace to Modules

A trace between unit tests and modules is provided in Table 3.

Table 3: Module Traceability

Module	**Test(s)**
M1	implemented by OS; no tests required
M2	external interface; no explicit testing; covered implicitly
M3	T11, T12, T13, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23
M4	data structure; no explicit testing; covered implicitly
M5	data structure; no explicit testing; covered implicitly
M6	T6, T7, T8
M7	T9, T10
M8	implemented by Python; no tests required
M9	T5

8 Code Coverage Metrics

The results of the code coverage analysis is shown in Figure 1. The target of 100% code coverage for the automated testing was successfully met.

9 Appendix

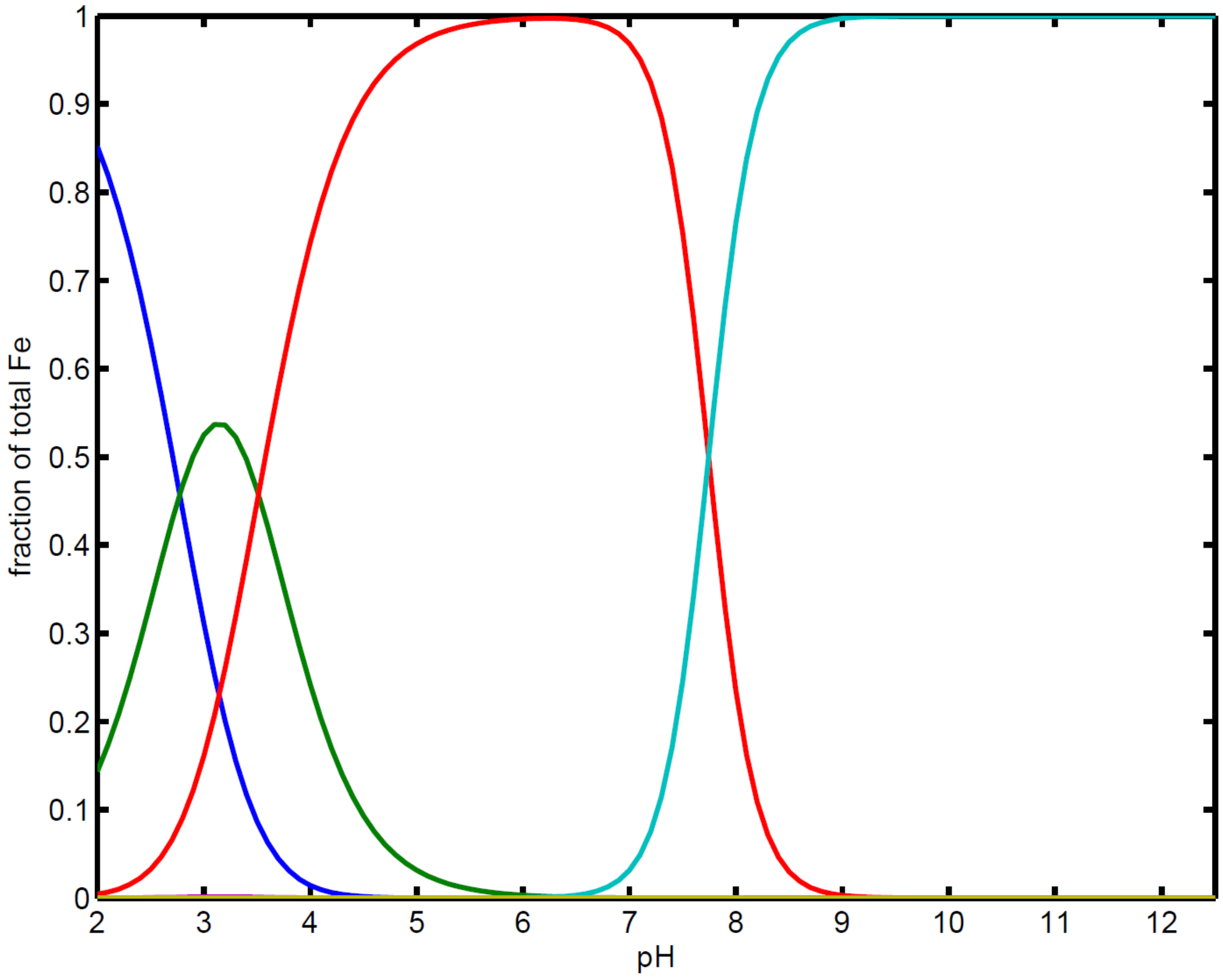


Figure 2: Dr. Smith's MATLAB implementation output for $FeOH_3$ system

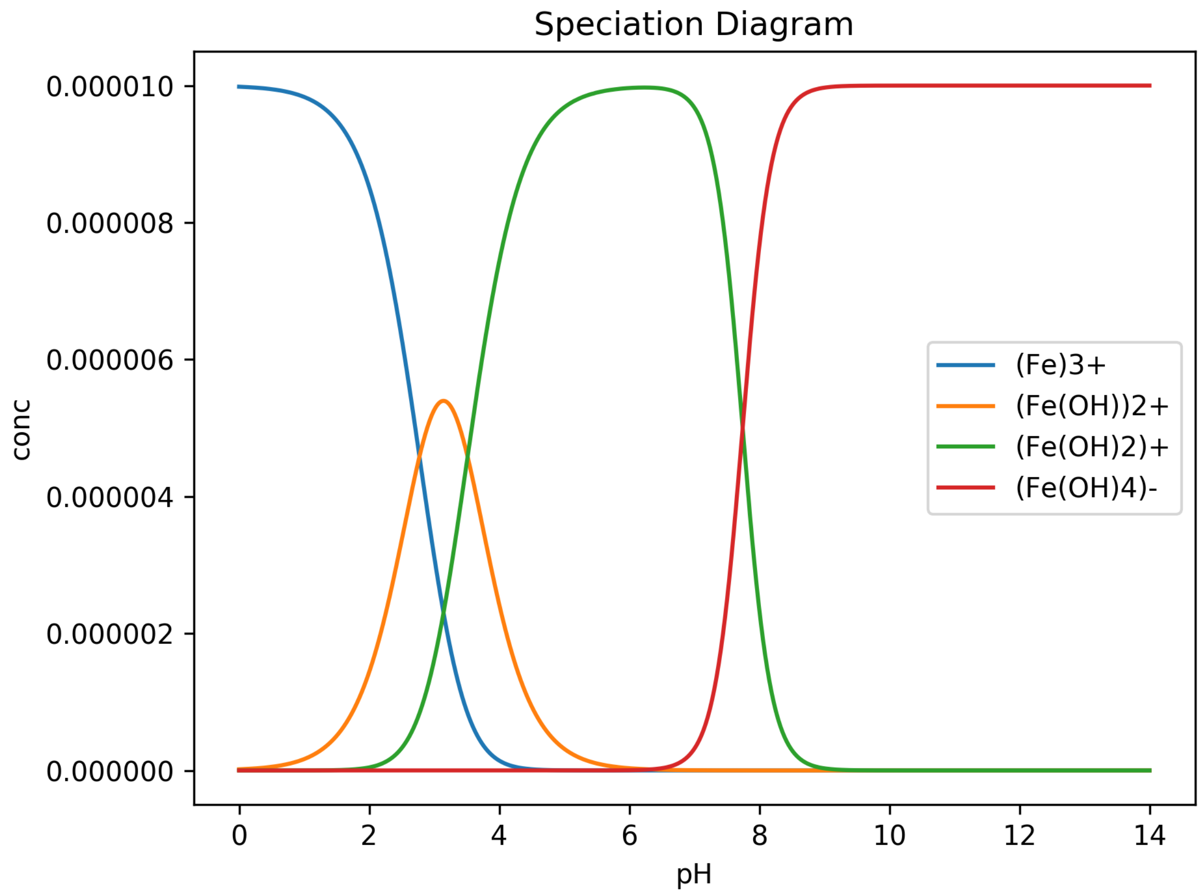


Figure 3: SpecGen output for $FeOH_3$ system