Building up to the equations from much F = ma (or F = mx’’) and making changing F (“the forcing term”)



|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
|  | Diagram  Description automatically generated | Diagram  Description automatically generated |

In ‘second\_law.py’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **The forcing term** | | **Equation** | **Result** | | **Comment** |
| Constant (linear) | | x’’ = F/m |  | | Constant acceleration (plot not working) |
| Constant times x (spring) | | x’’ = kx/m |  | |  |
|  | | x” = k(x-x0)/m |  | |  |
| Constant times x’, (damping) | | x’’ = cx’/m |  | |  |
| periodical | sin | x’’ = sin(nt)/m |  | |  |
|  |  | x’’ = (F-Asin(nt))/m |  | | Increasing amplitude and add F term |
|  | cos | x’’ = cos(t)/m |  | |  |
|  |  | x’’ = (F-Acos(nt))/m |  | |  |
| Combinations | | | | | |
| Spring mass damper system | | x” = (F-cx’-kx)/m |  | | <https://www.halvorsen.blog/documents/programming/python/resources/powerpoints/Mass-Spring-Damper%20System%20with%20Python.pdf> |
| Spring damper and sinusoidal term | | x” = (F-cx’-kx-sin(nt)/m |  | |  |
|  | |  | Changing the parameters |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective** | Applying SINDy-PI to spring-mass-damper system w/ pysindy (following ex 9 2d example) | | |
| **Test** | **Equation** | **Result** | **Comment** |
| Spring mass damper system | x” = (F-cx’-kx)/m | x0\_dot = 0.036 x0x1x1x1 + 0.041 x0x1x1x0\_dot + 0.025 x0x0x0x0\_dot + 0.069 x0x1x1x1x0\_dot + 0.008 x0x1\_dot + 0.285 x1x1\_dot + 0.026 x0x0x1\_dot + 0.185 x1x1x1\_dot + 0.050 x0x1x1x1\_dot + 0.016 x0x1x1x1x1\_dot + 0.013 x1x1x1x1x1\_dot + 0.024 x1x1x1x1x1x1\_dot  x1\_dot = 0.006 x0x1x1x1x0\_dot + -0.011 x0x0x0x0x0\_dot + 0.007 x0x0x0x0x0x0\_dot + -0.001 x0x0x0x0x0x0x0\_dot + -0.001 x1x1x1x1x1x1x0\_dot + 0.317 x0x1\_dot + 2.617 x1x1\_dot + -0.342 x0x1x1\_dot + -1.705 x1x1x1\_dot + 0.045 x0x0x0x0x1\_dot + -0.017 x0x0x0x0x0x1\_dot + 0.123 x1x1x1x1x1x1\_dot + 0.002 x0x0x0x0x0x0x1\_dot + 0.009 x1x1x1x1x1x1x1\_dot | Didn’t include the sin term (or exponential term)  Still very non sparse |
| Spring damper and sin term | x” = (F-cx’-kx-sin(nt)/m | x0\_dot = 0.940 x1 + 0.002 x0x0x0 + 0.007 x1\_dot + 0.036 x1x0\_dot + 0.003 x1x1\_dot  x1\_dot = -0.037 x0x1x1 + 0.011 x0x0x0x0 + -0.004 x0x0x0x0x0 + 0.032 x1x1x1x1x1 + -0.001 x0x0x0x0x0\_dot + -0.007 x1x1x1x1x1x1x0\_dot + 0.385 x0x1\_dot + 2.525 x1x1\_dot + -0.446 x0x1x1\_dot + -1.574 x1x1x1\_dot + 0.078 x0x1x1x1x1\_dot + 0.023 x0x0x0x0x1\_dot + -0.010 x0x0x0x0x0x1\_dot + 0.086 x1x1x1x1x1x1\_dot + 0.001 x0x0x0x0x0x0x1\_dot | Trying in pysindy (following ex 9 2d example)  Where x0\_dot = velocity, x1\_dot = acceleration  This library doesn’t include sin terms so equations won’t be right |
| Adding exponentials and sin terms to library | - | x0\_dot = 0.157 exp(x1) + -0.013 x1x1x1x1x1x1x0\_dot + 0.023 exp(x0)x0\_dot + 0.127 exp(x1)x0\_dot + 0.030 x0x1x1\_dot + 0.029 x1x1x1\_dot + -0.005 x1x1x1x1x1x1x1\_dot + 0.026 exp(x1)x1\_dot  x1\_dot = 0.005 x0x0x0x0 + -0.002 x0x0x0x0x0 + 0.002 x1x1x1x1x1x1x0\_dot + -0.008 exp(x1)x0\_dot + -0.100 x0x1x1\_dot + -0.896 x1x1x1\_dot + -0.020 x1x1x1x1\_dot + -0.013 x0x0x0x0x1\_dot + 0.003 x0x0x0x0x0x1\_dot + -0.001 x0x0x0x0x0x0x1\_dot + 0.061 exp(x0)x1\_dot + 0.694 exp(x1)x1\_dot | Includes exponentials terms but not sin terms?  Try w/ out exponentials |
| Removed exponential terms from the library | - | x0\_dot = 0.194 x1 + 0.002 x0x0 + 0.085 x1x1x1 + 0.009 x0x1x1x1 + 0.131 x0x0\_dot + 0.088 x1x0\_dot + 0.049 x0x0x0\_dot + 0.056 x1x1x0\_dot + -0.001 x0x0x0x0x0x0\_dot + 0.257 x1x1\_dot + 0.038 x1x1x1\_dot + 0.003 x1x1x1x1\_dot + 0.008 x0x1x1x1x1\_dot  x1\_dot = -0.001 x0x0x0x0 + 0.012 x0x1x1x1x0\_dot + -0.013 x0x0x0x0x0\_dot + 0.005 x0x0x0x0x0x0\_dot + -0.001 x0x0x0x0x0x0x0\_dot + -0.197 sin(x0x1)x0\_dot + 0.694 x1x1\_dot + -0.272 x0x1x1\_dot + 0.068 x0x0x0x1\_dot + -0.399 x1x1x1x1\_dot + 0.245 x0x1x1x1x1\_dot + 0.013 x0x0x0x0x1\_dot + -0.009 x0x0x0x0x0x1\_dot + 0.001 x0x0x0x0x0x0x1\_dot + 0.049 x1x1x1x1x1x1x1\_dot + 0.871 sin(x0x1)x1\_dot | Include sin terms but still very busy, increase sparsification term |
| Kept the library the same, in the optimiser function increased threshold from 10 to 100 | - | x0\_dot = 0.009 x0x0x0 + -0.001 x0x0x0x0x0 + 0.024 x1\_dot + 0.021 x0x0x0\_dot + 0.026 x0x0x0x0\_dot + 0.158 x0x1x1x1x0\_dot + -0.002 x0x0x0x0x0x0\_dot + 0.352 x1x1\_dot + 0.014 x0x1x1x1x1\_dot + 0.043 x1x1x1x1x1\_dot + 0.088 x1x1x1x1x1x1\_dot  x1\_dot = -0.010 x0x0x0x0x0\_dot + 0.004 x0x0x0x0x0x0\_dot + 0.065 x0x0x0x1\_dot + 0.083 x0x1x1x1x1\_dot + -0.004 x0x0x0x0x0x1\_dot + 1.017 sin(x0x1)x1\_dot | Increase sparsification threshold even more   * If it still looks just as busy after this revert threshold to 10 and look at other parameters in the optimiser function |
| Threshold from 100 – 300 |  | x0\_dot = 0.180 x1\_dot + 0.029 x0x0x0x0\_dot + 0.169 x0x1x1x1x0\_dot + -0.002 x0x0x0x0x0x0\_dot + 0.026 x1x1\_dot + 0.176 x1x1x1x1x1x1\_dot  x1\_dot = -0.010 x0x0x0x0x0\_dot + 0.004 x0x0x0x0x0x0\_dot + 0.066 x0x0x0x1\_dot + 0.034 x0x1x1x1x1\_dot + -0.004 x0x0x0x0x0x1\_dot + 0.025 x1x1x1x1x1x1x1\_dot + 0.906 sin(x0x1)x1\_dot | More significant difference in the model and actual solution (looking at graph)  Try another term in the optimiser to change? |
| Threshold from 300 – 1000 |  | x0\_dot = 0.025 x0x0x0x0\_dot + 0.119 x0x1x1x1x0\_dot + -0.001 x0x0x0x0x0x0\_dot + 0.026 x1x1x1x1x1x1x0\_dot + 0.111 x1x1x1x1x1x1x1\_dot  x1\_dot = -0.015 x0x0x0x0x0\_dot + 0.005 x0x0x0x0x0x0\_dot + 0.001 x1x1x1x1x1x1x0\_dot + 0.009 x0x0x0x0x1\_dot + 0.070 x1x1x1x1x1x1x1\_dot + 0.541 sin(x0x1)x1\_dot | Model definitely significantly wrong  Equations are sparser but still too many terms and not the right ones |
| Revert threshold, new library that only has the terms in the equations |  |  |  |