即时编译(just-in-time compilation): JIT

是一种执行计算机代码的方法,这种涉及执行过程中而不是执行之前进行编译。JIT编译将编译代码的速度与解释的灵活性、解释器的开销以及额外的编译开销结合起来。

编译型语言: 编写的源程序需要经过编译,汇编和链接才能输出目标代码,然后机器执行目标代码 ⇒ C / C++

解释型语言: 翻译器不产生目标机器代码,产生易于执行的中间代码,中间代码的解释是由软件支持的。⇒ Js, Ruby, Perl

特殊的 Java: Java也需要编译,也需要解释。直接编译为字节码,在Java虚拟上解释执行字节码 ⇒ Python 与之类似

编译型语言　　非常动态的解释型语言

- 结论: C语言的执行效率比Python强太多了。
  - ※ 对换算耗时 C语言耗时大约比Python少两个数量级左右(10~100)
    - ↳ 严格上,很难直接转换耗时,相同算法消耗越长的时间越能体现出C语言的高效性。
- 原因: 不完全的类型意味着两种语言在机器上实现去的路径大不相同。
  - Python为了灵活性牺牲了太多的效率
  - ⇒ Python的解释器拉跨。官方提供,最大范围使用的解释器CPython效率不佳。(Pyng中的镜像,使用的是CPython)
    也存在其他类型的解释器如PyPy, GraalPython 但大多有所限制。

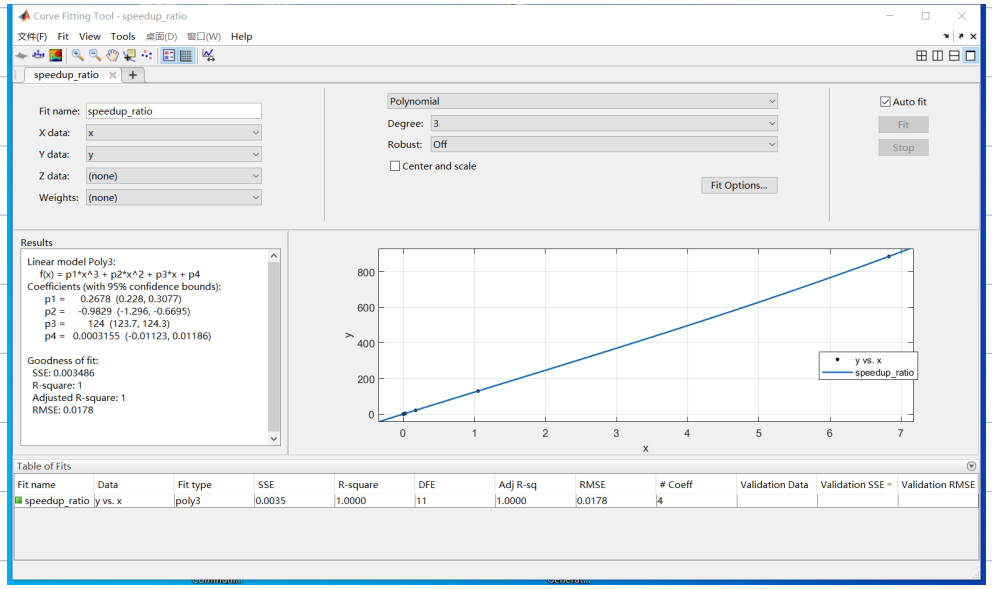
以下总结一些题外话:

① Python的执行效率还有没有办法提升?

不太可能针对整个语言进行优化。当前存在的优化方法(numba)(JIT)也仅是对Python的子集进行了优化,其语义太过于动态。另外Python调用的包还大多是基于C / C++开发的。借助加速手段,可以使部分Python的执行效率追平C++。

Q Python的效率很低，为什么受众如此广？

开源，不源于任何一家企业。面向新手友好 → 社区丰富。



y轴数据：为 Python 原代码执行的耗时

x轴数据：为 numba 加速 Python 代码的执行时间（加 C++ 的效率）

⇓

拟合三次函数，pynq上执行比对耗时 0.01s

由该曲线得到 C++ 语言执行耗时 7.81×10^{-5} s

⇕

仿真耗时 3×10^{-5} s

⇕

加速比 2.6 左右