双酚 BPTK 报告文档

许敬然

当前可用于展开的原始资料与基本情况说明

当前我共从章学长处拿到一份微分方程组形式的 PBTK 模型，两份模型求解的 R 语言代码，包含各类参数。其中有两版不同的模型。

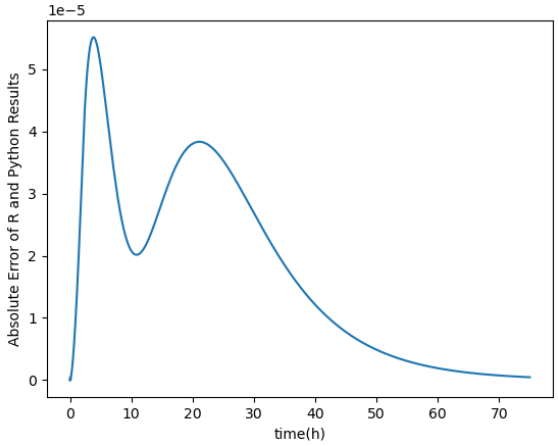
较早期的一版模型出于章学长的论文，形式是一份粘贴在 word 文档的 R 代码，里面用 R 语言中 ode() 求解器求解了 63 个变量关于时间的曲线。其中只有 26 个变量是实际所需求解的对象，它们存在有实际意义的关于时间的导数，剩余的变量只作为计算中间量。这些中间量可能并不存在关于时间的导数，甚至有些变量并无实际意义，只是为了表示更方便，作为实际变量的导数表达式中的一部分出现。但是为了保证在求解器中的计算顺序准确，这些中间变量以某个虚构函数关于时间的导数信息的身份出现在求解器里，同其他导数信息共同被计算、求解。例如涂抹时期内的用药剂量 dose，实际上它在涂抹时期为一个正的常数，涂抹期过去后始终为 0，但在代码中，它作为中间变量以 “Rdose” 的名字出现在了求解器的导数信息部分中。

此论文除了共享了部分模型求解代码，还共享了受试志愿者的部分生理参数与尿液中非结合型双酚 S 的浓度与双酚 S 和双酚 S-g 的总浓度。代码里提供的参数与方程都来源于 BPS(双酚 S)(大部分方程与 BPA(双酚 A) 在人体内的情况相同)。其对应的实验情况是在四天内每一天同一时间令受试志愿者的手指与 TP(Thermal-Paper 热敏纸) 持续接触，1 分钟 (涂抹时长) 后停止触摸，此时手指上有残留的 BPs(双酚类物质)。130 分钟 (暴露时长) 后用特殊试剂彻底清洗受试手指，此时表皮储仓内的 BPs 认为已清零，对应代码中的变量 AWELL 在 13/6(h) 处及后续的值与导数值置为零。与此同时，在受试期内，每 260 分钟取得一次受试者的尿液并得到其中非结合型双酚 S 的浓度与双酚 S 和双酚 S-g 的总浓度。

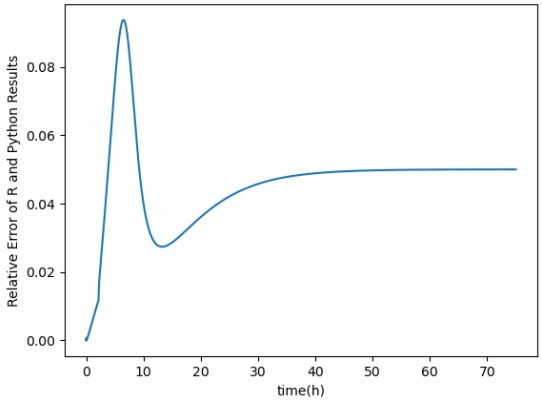
后面一版模型都是关于 BPA(双酚 A) 的，形式为一个有 28 个微分方程的方程组，其中除了第 23、24 个方程是非线性外，其他都是线性的。模型中的部分生理生化参数并未延续上述论文中的数值，而是有所更改。与之联系的代码是陈老师和章学长共同讨论过的带指数 exp 格式的模型求解代码，由于讨论搁置，该求解代码并不是最终完善版本。该版本的模型对应的实验情况有两种，其中第一种与上述论文中的实验方法相同，都是令手指与 TP 接触，涂抹时间都是 1 分钟，暴露时长 (从开始涂抹到清洗) 改为 60 分钟。第二种是 PCP(Personal-Care-Product 个人护理产品) 手臂暴露，四天内每 12 小时令受试者手臂涂抹 PCP 一分钟，留置 6 小时后彻底清洗，表皮储仓内的 BPs 清零。由于受试部位不同，两种暴露实验所使用的皮肤参数有多出不同。该研究没有包含真实受试者数据。

早期模型求解代码 (BPS) 的 Python 复现与结果验证

早期代码使用的部分人体生理参数来源于真实的受试者数据 (受试者编号 A)，皮肤参数的设定基于手指皮肤。将链接文件的读取和参数平移至 Python 中，将变量与导数信息输入至 scipy.integrate 包

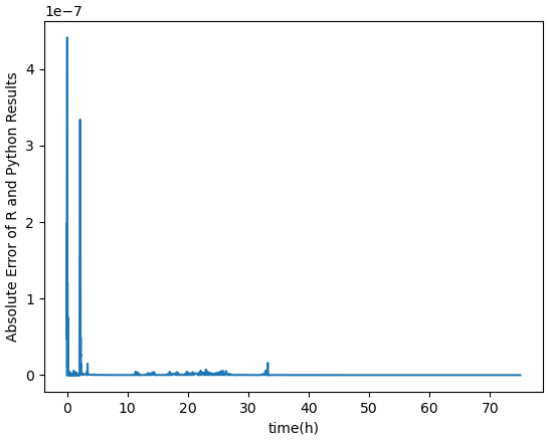


(a) 两种结果的绝对误差曲线

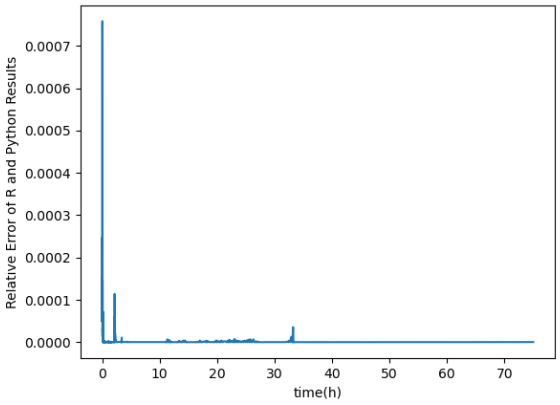


(b) 两种结果的相对误差曲线

图 1: 源代码与复现代码 A 计算的血药浓度对比



(a) 两种结果的绝对误差曲线



(b) 两种结果的相对误差曲线

图 2: 源代码与复现代码 B 计算的血药浓度对比

中的的 odeint 求解器内求解。求解的时间序列为 arange(0, 75, 0.005)，从零到第 75 小时，每 18 秒为一个节点，共 15000 个节点。所有变量的初值都为 0。之后将 R 代码中计算模型得到的结果 (15001*64 格的 csv 文件，其中第一列为时间序列) 读取至 Python 中与 Python 的计算结果对比。

在求解时，最初我把源代码中 63 个变量精简为了 26 个变量，并未把原模型的中间变量放入求解器的导数信息中，而是把中间变量的计算公式放在了导数信息之外。这样的操作会导致在时间序列走完一步，26 个变量都更新完数值后，中间变量才被计算，设有变量的计算公式为 $u_t = f(u_{t-1}, \theta_t)$，其中 θ 是中间变量, 计算公式设为 $\theta_t = g(v_t)$，如果按照我刚刚的这种操作方式，中间变量在求解迭代一轮结束后单独计算，u_t 的计算需要使用到 θ_t 的信息，但是能拿到最新的 θ 只能是 θ_{t-1}，故误差出现了。该代码与源代码血浆内 BPS 浓度随时间变化曲线的对比结果如图1。

可以发现绝对误差曲线的趋势与原曲线类似且较为光滑，相对误差的最大值超过了 8%。察觉到错误后，我又将中间变量作为导数信息放在了求解器中，得到的对比结果如图2。可以看到这份代码与源代码得到的结果几乎是相同的。

28 方程求解模型 (BPA) 的 Python 复现与问题

我拿到的基于 28 方程模型的 R 代码并没有使用求解器 (带指数的迭代格式)，在复现时，我使用了求解器求解 28 方程，得到的结果与 R 代码大相径庭。我又试着用 BPS 的模型代码求解 (上一段有 63 个变量的求解器)，部分参数改成 BPA 的参数 (BPA 和 BPS 在人体内的代谢高度相似，只有部分参数不同)，得到的结果也与 28 方程模型相差极大，三种方法得到的结果对比如图3。

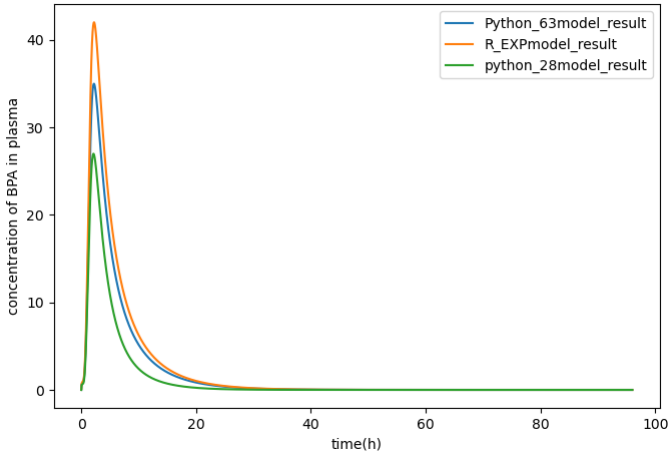


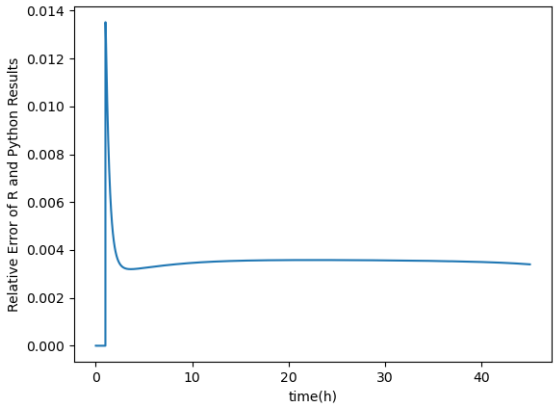
图 3: 三种代码的结果对比

用求解器直接求解 28 方程组的结果与 R 代码中的指数格式不同是可以理解的。因为第 23 与 24 个方程并非线性方程，而在 R 代码中对这一点做了求解上的近似：在用积分因子法导出这两个方程的计算格式时，将非线性部分视为了常数部分，只使用了线性部分，像其他线性方程一样导出了相同的带指数的格式。这样的近似带来了偏差是合理的。

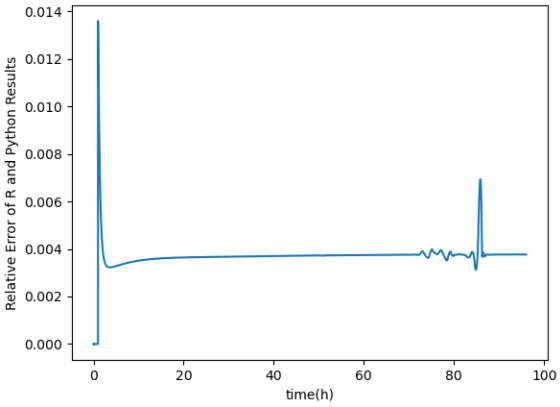
但 28 方程模型实际上是对 63 变量求解法的整理，去除了绝大部分中间变量，将所求变量精简到了 28 个。但用求解器直接求解这 28 个方程是会出现错误的，例如第 24 个变量 Aliver，代表了肝脏中化学品含量，按理说应该始终是非负值，但是求解得到的结果一直是非正的。我试着用 R 中的求解器 deSolve 包内的 ode() 求解 28 方程组，得到的结果 (血浆药浓度、肝脏药浓度) 都与 Python 内求解器得到的结果高度相似，结果如图4。

可以看到在求解 28 方程组时，R 中求解器也出现了得到异常负值的情况。且根据两种求解器得到的血药浓度的高度一致可以断定，负值错误的发生与求解器本身没有关系，而是 28 方程组本身出现了问题。用 63 变量模型求解得到的肝脏药浓度是正值，且看起来和 28 方程模型求得的肝脏药浓度的绝对值很接近，事实上，这二者间的相对误差很高。部分结果如图5。

28 方程组需要进一步讨论与解析，目前的版本得到的血药浓度的变化趋势是正确的，但由于缺少一个正确的对照，并不清楚实际数值是否在可接受范围之内。同时，该版本得到的肝脏药浓度曲线是错误的，而基于 28 方程组得到的带指数的格式计算出的肝脏药浓度是正的。是否由于第 23,24 个方程的非线性部分造成两个结果的不同需要进一步探究。

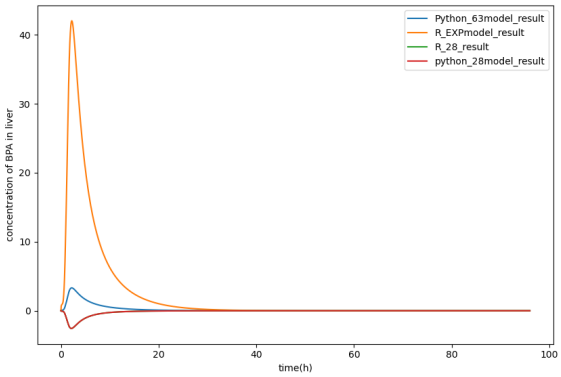


(a) 两种血药浓度的相对误差

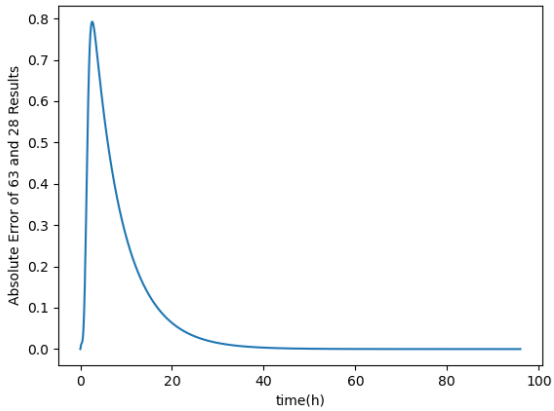


(b) 两种肝脏药浓度的相对误差

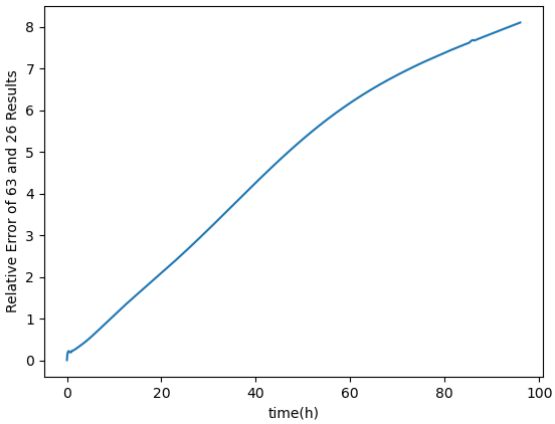
图 4: R 与 Python 求解 28 方程计算结果对比



(a) 几种方法得到的肝脏药浓度曲线



(b) 28 法与 63 法得到的肝脏药浓度的绝对值的绝对误差



(c) 28 法与 63 法得到的肝脏药浓度的绝对值的相对误差

图 5: 不同结果得到的肝脏药浓度结果对比

参数优化的 Python 复现 (基于 BPS 模型)

在后续与章学长的交流中，我拿到了一份基于 BPS 模型的参数优化代码。本文档的第一部分提到过，在 BPS 的人体 BPTK 模型研究中对受试者每隔 4.3333 小时采一次尿，以测量尿液中的未结合 BPS 含量和 BPS&BPS-g 的总含量。他们的优化参数的做法即是，使用 R 中的 optim() 函数在三参数的情况下最小化损失函数 (三个参数：DSC< 角质层中的有效扩散系数 >、PFO< 毛囊的渗透系数 >、u1< 由脱屑而向皮肤表面转移的速度 >)。其中损失函数是真实测量数据的时间序列和模型计算出的与之对应时间节点的尿药浓度之间的 SSE(误差平方和)。有效受试者共有四位，损失函数中的 SSE 是四名受试者对应 SSE 的总和。模型计算结果在对应采尿时间附近没有做平滑。

我首先准备复现这个优化算法。最开始，我选择了使用 scipy.optimize 包中的 minimize() 函数来最小化损失函数，尝试了函数中的许多 methods，但并不奏效。接下来换了另一个适用于调整超参数的优化函数 hyperopt.fmin()，效果很显著