

神经网络解 Lorenz63 方程

MG21210021 李庆春

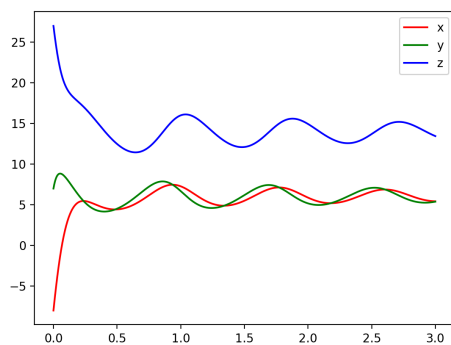
2023 年 3 月 20 日

1 Lorenz63 方程简述

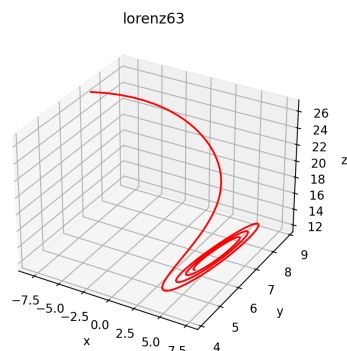
$$\begin{cases} \frac{dx}{dt} = \sigma(y - x) \\ \frac{dy}{dt} = x(\rho - z) - y \\ \frac{dz}{dt} = xy - \beta z \end{cases} \quad (1)$$

2 非混沌情形

$\rho = 15, \sigma = 10, \beta = \frac{8}{3}$ 初值: $[-8., 7., 27.]$, RK 方法的数值解如图 1:



(a) lorenz63 二维视图.



(b) lorenz63 三维视图.

图 1: lorenz63-RK

2.1 不带观测的神经网络

损失函数使用均方误差 (MSE), 网络结构: $[1, 32, 32, 3]$, 初值: $[-8, 7, 27]$, 迭代轮数: 500000, 对训练数据进行正规化, 关键代码如下:

1

```
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
```

```

2         min_lr=1e-7, mode='min', factor=0.5, patience=50000, verbose=True)
3         self.loss_func = nn.MSELoss(reduction='mean')
4
5         layers = [1, 32, 32, 3]
6
7         total_points = 300
8         x_lb = torch.tensor(0.)
9         x_ub = torch.tensor(1.24)
10
11        rho = torch.tensor(15.0)
12        sigma = torch.tensor(10.0)
13        beta = torch.tensor(8.0 / 3.0)
14
15        x_train_bc = torch.tensor([[0.]])
16        y_train_bc = torch.tensor([[-8., 7., 27.]])
17
18        epochs = 500000

```

通过调整区间长度发现，最多可训练出 $[0, 1.24]$ 的结果，误差在 10^{-4} 量级，如图 2：

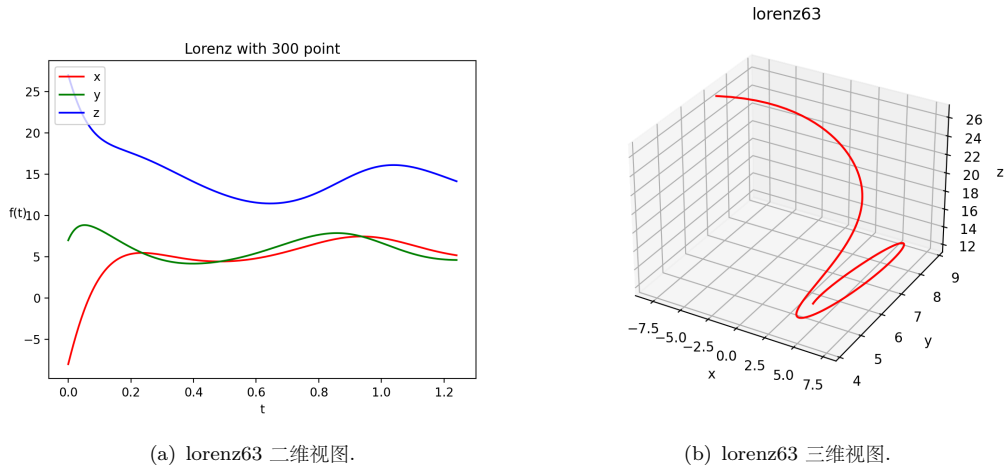
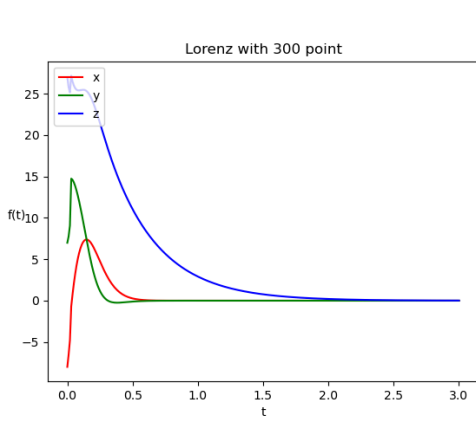


图 2: lorenz63-DNN

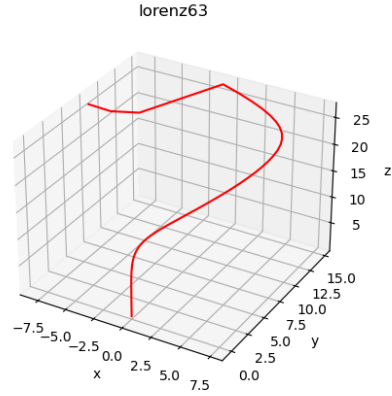
如果区间较大，则训练不出想要的结果，神经网络会趋于零，如图 3 展示的是 $[0, 3]$ 的训练结果：

3 混沌情形

$\rho = 28, \sigma = 10, \beta = \frac{8}{3}$ ，初值 $[-4, 7, 15]$ ，RK 方法的数值解如下图：（暂时没贴）



(a) lorenz63 二维视图.



(b) lorenz63 三维视图.

图 3: lorenz63-DNN

3.1 整数阶 Lorenz 方程

接下来用神经网络训练，离散步长 $h=0.005$ ，区间 $[0, 1.5]$ ，观测值如下：

t	x	y	z
0.0000	-4.0000	7.0000	15.0000
0.5000	5.6789	-2.6593	33.0093
1.0000	-12.0143	-17.4640	24.3640
1.500	-1.1838	-1.6864	15.08598

3.2 分数阶 Lorenz 方程

3.2.1 Caputo 导数及数值逼近

Caputo 分数阶导数的定义：

$${}_a D_t^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t (t-\tau)^{n-\alpha-1} f^{(n)}(\tau) d\tau \quad (n-1 \leq \alpha < n). \quad (2)$$

当 $a=0$ 时，简记为 D^α 。

分数阶微积分的差商逼近格式可以写成如下统一的形式：

$$D^\alpha f(t_n) \approx h^{-\alpha} \sum_{k=0}^N c_{n,k} f_k \quad (3)$$

L1 算法（即 $0 \leq \alpha < 1$ ，上式中 $N=n$ ）：

$$c_{n,k} = \frac{1}{\Gamma(2-\alpha)} \begin{cases} -c_{n-1}, & k=0; \\ c_{n-k} - c_{n-k-1}, & 1 \leq k \leq n-1; \\ 1, & k=n; \\ 0, & else. \end{cases} \quad (4)$$

$$(D_t^\alpha f(t_n))_{L1} = \frac{1}{\Gamma(2-\alpha)h^\alpha} \sum_{k=0}^{n-1} c_k(f_{n-k} - f_{n-k-1}) \quad (5)$$

其中, $c_l = (l+1)^{1-\alpha} - l^{1-\alpha}$, fPINN 论文中另一种等价表述如下:

$$\begin{aligned} \frac{\partial^\gamma \tilde{u}(\mathbf{x}, t)}{\partial t^\gamma} &\approx \frac{1}{\Gamma(2-\gamma)(\Delta t)^\gamma} \\ &\left\{ -c_{\lceil \lambda t \rceil - 1} \tilde{u}(\mathbf{x}, 0) + c_0 \tilde{u}(\mathbf{x}, t) + \sum_{k=1}^{\lceil \lambda t \rceil - 1} (c_{\lceil \lambda t \rceil - k} - c_{\lceil \lambda t \rceil - k - 1}) \tilde{u}(\mathbf{x}, k\Delta t) \right\}, \end{aligned} \quad (6)$$

$0 < \gamma < 1$

时间步长 $\Delta t = t/\lceil \lambda t \rceil \approx 1/\lambda$, $\lceil \cdot \rceil$ 是向上取整函数, 表示离散化后的区间数, 相当于 L1 算法公式中的 n ; 常量因子 λ 决定着步长, 如 $\lambda = 200$, 则意味着把单位 1 长度划分为 200 个小区间。由于计算机在运算时, 存在舍入误差, 所以 $c_0 \tilde{u}(\mathbf{x}, n\Delta t)$ 写成 $c_0 \tilde{u}(\mathbf{x}, t)$ 。

(这里有疑问, $t=0$ 的导数是不是没法计算)

所以, 可先对每个训练点计算出其时间划分, 然后拼成一个大向量。

如果不引入观测, 训练结果的趋势是趋于 0, 不符合预期;

3.2.2 分数阶常微分方程数值解法

(直接法) 考虑齐次初值条件的分数阶微分方程:

$$\begin{cases} \frac{\partial^\alpha y(t)}{\partial t^\alpha} = f(t, y(t)), & t \in [0, T], \\ y^{(k)}(0) = 0, & k = 0, 1, \dots, m-1. \end{cases} \quad (7)$$

则直接应用分数阶导数的一般差商逼近公式得:

$$h^{-\alpha} \sum_{k=0}^N c_{n,k} y_k = f(t_n, y_n), \quad n = 0, 1, \dots, [t/h] \quad (8)$$

上式左边为 y 在 t 处的 α 阶导数, 注意此时将 $[0, t]$ 分成了 n 份, 每份长度为 h , t 对应 t_n , $y(t_n)$ 对应 y_n 。则上面方程组可以按下面的方式逐点计算:

$$y_N = \frac{h^\alpha}{c_{n,N}} f(t_n, y_n) - \frac{1}{c_{n,N}} \sum_{k=1}^{N-1} c_{n,k} y_k, \quad n = 1, \dots, [T/h] \quad (9)$$

其中, $N=n$ (对应到 G1 算法、D 算法、L1 算法、线性多步法) 或 $N=n+1$ (对应到 G2 算法、L2 算法)。上式中, 由于 $y_0 = 0$, 所以求和项从 $k=1$ 开始, 对于一般的 L1 算法, 可写成如下形式:

$$\begin{aligned} y_n &= \frac{h^\alpha}{c_{n,n}} f(t_n, y_n) - \frac{1}{c_{n,n}} \sum_{k=0}^{n-1} c_{n,k} y_k, \quad n = 1, \dots, [T/h] \\ &= \Gamma(2-\alpha) h^\alpha f(t_n, y_n) - \sum_{k=1}^{n-1} (c_{n-k} - c_{n-k-1}) y_k + c_{n-1} y_0 \\ &= \Gamma(2-\alpha) h^\alpha f(t_n, y_n) - \sum_{k=1}^{n-1} [(n-k+1)^{1-\alpha} - 2(n-k)^{1-\alpha} + (n-k-1)^{1-\alpha}] y_k + \\ &\quad [n^{1-\alpha} - (n-1)^{1-\alpha}] y_0 \end{aligned} \quad (10)$$

3.2.3 简单的例子

已知正弦、余弦函数的整数阶微分表达式分别为

$$\frac{d^k}{dt^k}[\sin at] = a^k \sin(at + \frac{k\pi}{2}), \quad \frac{d^k}{dt^k}[\cos at] = a^k \cos(at + \frac{k\pi}{2}) \quad (11)$$

由 Cauchy 积分公式可以证明, 对于分数阶的微分方程来说, 当 k 为分数时, 上述公式仍然成立。所以考虑如下分数阶微分方程:

$$\begin{cases} \frac{d^\alpha u}{dt^\alpha} = \sin(t + \frac{\alpha\pi}{2}), & 0 < \alpha < 1 \\ u(0) = 0 \end{cases} \quad (12)$$

(这里暂时不知道解析解, 所以作废)

对于 $f(t) = t^\lambda, \lambda > -1$, n 为大于 α 的最小整数, 则其 α 阶 Caputo 导数为:

$$\begin{aligned} D^\alpha t^\lambda &= \frac{1}{\Gamma(n-\alpha)} \int_0^t (t-\tau)^{n-\alpha-1} \frac{d^n \tau^\lambda}{d\tau^n} d\tau \\ &= \frac{1}{\Gamma(n-\alpha)} \int_0^t (t-\tau)^{n-\alpha-1} \lambda(\lambda-1)\cdots(\lambda-n+1) t^{\lambda-n} d\tau \\ &= \frac{\Gamma(\lambda+1)}{\Gamma(n-\alpha)\Gamma(\lambda-n+1)} \int_0^t (t-\tau)^{n-\alpha-1} t^{\lambda-n} d\tau \\ &= \frac{\Gamma(\lambda+1)}{\Gamma(n-\alpha)\Gamma(\lambda-n+1)} t^{\lambda-\alpha} \int_0^1 (1-\tau)^{n-\alpha-1} t^{\lambda-n} d\tau \quad (\text{substitution}) \\ &= \frac{\Gamma(\lambda+1)}{\Gamma(n-\alpha)\Gamma(\lambda-n+1)} t^{\lambda-\alpha} B(n-\alpha, \lambda-n+1) \\ &= \frac{\Gamma(\lambda+1)}{\Gamma(n-\alpha)\Gamma(\lambda-n+1)} t^{\lambda-\alpha} \frac{\Gamma(n-\alpha)\Gamma(\lambda-n+1)}{\Gamma(\lambda-\alpha+1)} \\ &= \frac{\Gamma(\lambda+1)}{\Gamma(\lambda-\alpha+1)} t^{\lambda-\alpha} \end{aligned} \quad (13)$$

这和 R-L 导数一致, 这是由于 $D^n t^\lambda$ 在区间 $[0, +\infty)$ 上连续, 且 $D^k f(0) = 0, k = 0, 1, \dots, n-1$, 所以 D^n 和 $D^{-\nu}$ 可交换。

例子: 考虑 $y = t^3$, 则对应的分数阶微分方程为:

$$\begin{cases} D^\alpha t^3 = \frac{6}{\Gamma(4-\alpha)} t^{3-\alpha} \\ y(0) = 0 \end{cases} \quad (14)$$

写成隐式的:

$$\begin{cases} D^\alpha u = \frac{6u}{\Gamma(4-\alpha)} t^{-\alpha} \\ u(0) = 0.000001 \end{cases} \quad (15)$$

数值计算结果如图 4: (这里有疑问, 时间是不是一定要从 0 开始, 变成时发现如果不是 0 作为离散的初始点, 则得不到结果)

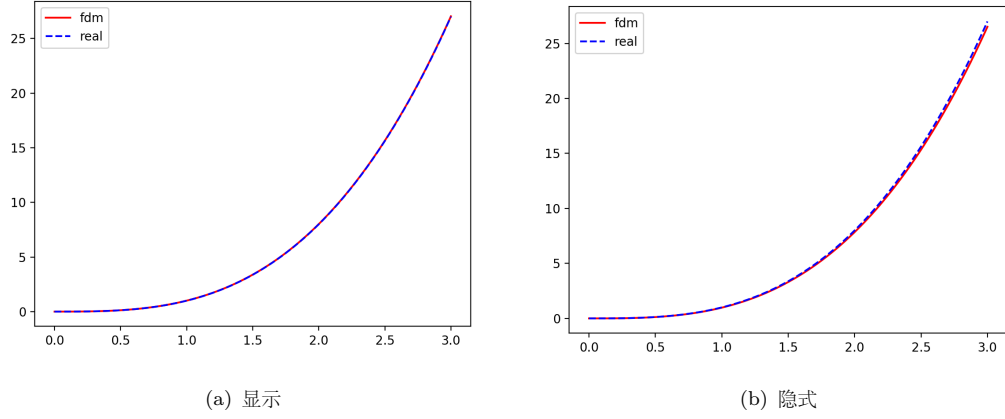


图 4: ODE-FDM

3.2.4 分数阶 Lorenz63 方程

$$\begin{cases} \frac{d^\alpha x}{dt^\alpha} = \sigma(y - x) \\ \frac{d^\alpha y}{dt^\alpha} = x(\rho - z) - y \\ \frac{d^\alpha z}{dt^\alpha} = xy - \beta z \end{cases} \quad (16)$$

取 $\alpha = 0.99$ ，离散步长 $h=0.005$ ，区间 $[0, 3]$ ，隐式收敛阈值 $\epsilon = 0.001$ ，初值 $[-4., 7., 15.]$ ，数值计算结果如图 5:

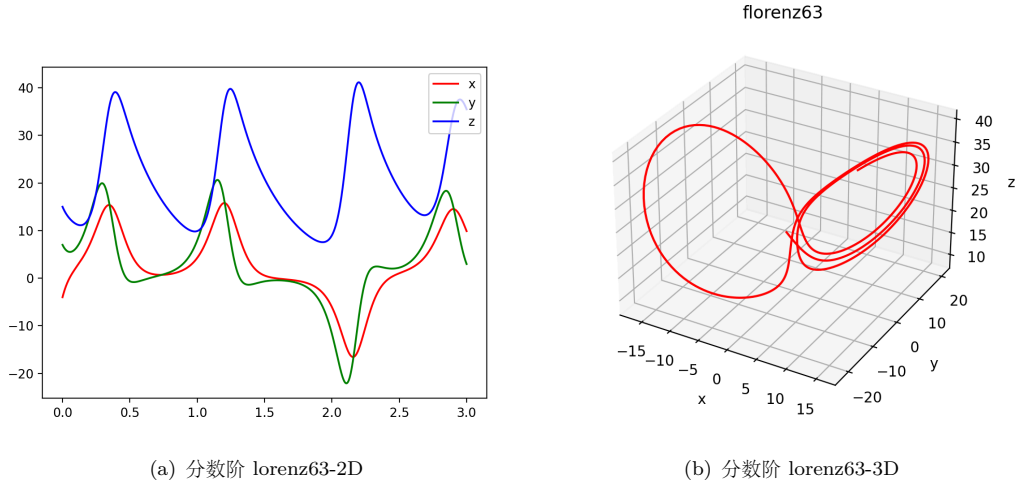
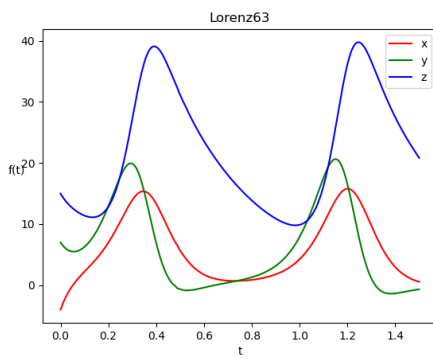


图 5: frac-lorenz63-FDM

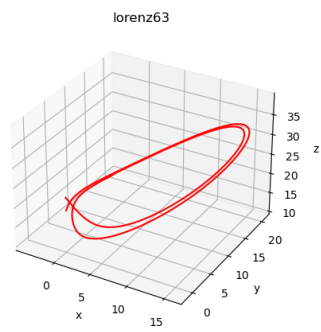
接下来用神经网络训练，损失函数使用 MSE，先用参数 $\alpha = 0.99$ ，离散步长 $h=0.005$ ，区间 $[0, 1.5]$ ，观测值如下:

t	x	y	z
0.0050	-3.4659	6.7226	14.6728
0.2500	10.4394	17.7745	17.9601
0.5000	5.6236	-0.6432	31.2168
0.7500	0.7111	0.8086	15.8068
1.0000	4.2548	7.9222	9.8927
1.2500	13.9364	7.5091	39.7435
1.5000	0.5546	-0.6693	20.8581

训练结果如图 6，符合预期：



(a) 分数阶 lorenz63-2D



(b) 分数阶 lorenz63-3D

图 6: frac-lorenz63-FDM

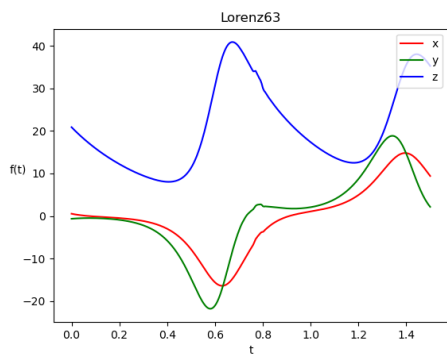
继续训练 [1.5, 3]，观测数据如下，注意此时 $t=0$ 对应的是 $t=1.5$ ：

t	x	y	z
0.0050	0.4947	-0.6466	20.5689
0.2500	-0.7559	-1.3882	10.706
0.5000	-7.8296	-14.6883	11.2953
0.7500	-7.6905	1.1865	34.9326
1.0000	1.0532	2.0665	17.3576
1.2500	7.2898	12.7107	14.2827
1.5000	9.4016	2.1509	35.2638

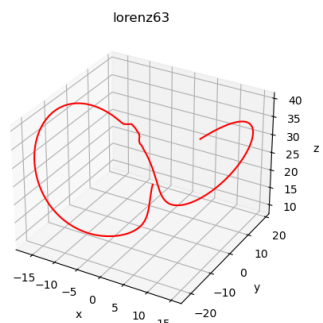
训练结果如图 7，除个别点尖锐突变，根据几次训练，突变点在 [0.4,0.8] 之间，有时是个小尖角，其他部分符合预期：所以继续对 [0.4,0.8] 增加一些观测值，如下：

t	x	y	z
0.4000	-3.0048	-5.7771	8.0258
0.6000	-15.5885	-21.0788	30.5955
0.7000	-12.4916	-3.5173	39.8045
0.8000	-3.9029	2.2364	29.9654

得到的训练结果如图 8：继续增加观测值，变为每 0.1 的长度，引入一个观测值，训练结果如

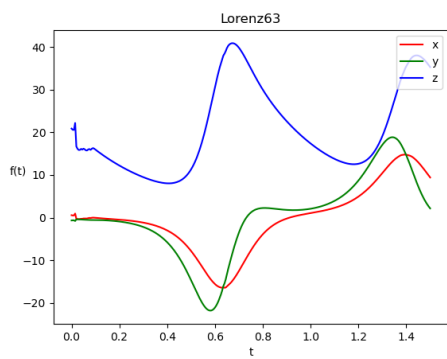


(a) 分数阶 lorenz63-2D

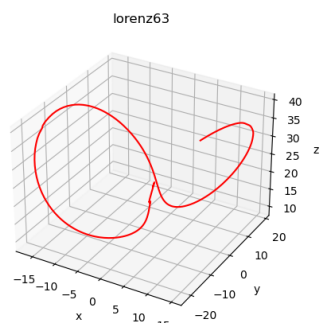


(b) 分数阶 lorenz63-3D

图 7: frac-lorenz63-FDM



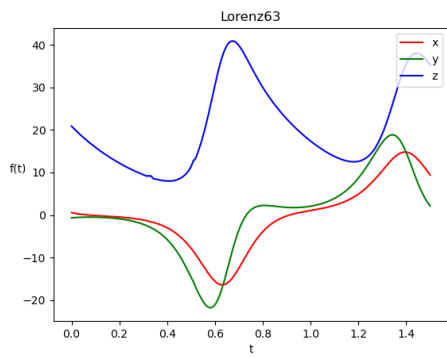
(a) 分数阶 lorenz63-2D



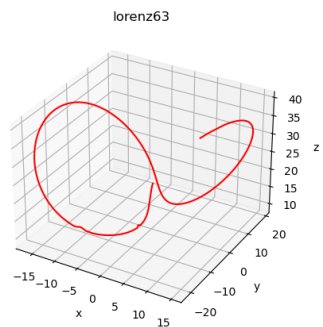
(b) 分数阶 lorenz63-3D

图 8: frac-lorenz63-FDM

图 9:



(a) 分数阶 lorenz63-2D



(b) 分数阶 lorenz63-3D

图 9: frac-lorenz63-FDM