NUMERICAL ASPECTS FOR NONLOCAL PROBLEMS

POLYMATH 2024

Abstract. Traditional ways to solve differential equations (DEQs) have had remarkable results, such as the forward Euler, improved Euler, and Runge Kutta methods. However, with the development of science and technology, sometimes it's really hard to get the gradient in reality. The only thing we can do is sample the objective function to estimate the gradient. Nowadays, in the Natural Language Processing (NLP) field, there are lots of derivative-free situations (DALL · E, GPT-4, GPT-4o, etc.), which means that we cannot get the true gradient from the APIs except for the object function's value. So how to optimize the loss function in the black box tuning situation becomes more intractable because since the gradient of the function cannot be obtained, the gradient descends and the backpropagation method no longer works. In this paper, we want to use the nonlocal derivative to estimate the true derivative (1-dimensional) which gives us a new perspective on how to take cognitive derivatives, and we will use this method to solve DEQs. At the end of the paper, we can realize this algorithm gives us a whole new perspective and a way to calculate gradients, it may have a large application in black box optimizing, black box tuning in NLPs, and solving differential equations with unknowable functions.$-Zhekai$

1. Participants

- Zhekai Liu (Liu_Zhekai@outlook.com)
- Emrys King (egka2021@mymail.pomona.edu)
- Dianlun (Jennifer) Luo (dl3572@columbia.edu)
- Anders Bahrami (andersbahrami@gmail.com)
- Diego Rubio (diegorubio387@gmail.com)

References

2. Meetings

Zoom meeting link: https://pomonacollege.zoom.us/j/82038666220
Meeting ID: 820 3866 6220 Passcode: nonlocal
PR: Please see my comments below in blue. Once you address them, I will remove them. Also, please feel free to leave questions for us in the writeup.

Meeting 07/11:

- Plan to meet at least weekly on Thursdays at 11 am CT, At each meeting share what you're thinking about or planning to work on. PR: Excellent idea to meet regularly as a group.
- Write down any thoughts, questions you have, research directions, etc. in overleaf
- Join Matlab shared drive and consider matlab onramp: `https://drive.mathworks.com/sharing/2f0e040f-2275-4816-af05-35c892720070`

TODO:
Write basic Euler's method programs for solving systems of DEQs (think about going beyond Euler's method, Runge-Kutta, etc.)

Think about assisting other subtopics in answering their questions, (also consider investigating by hand first and then computationally to build intuition)

Meeting 07/18:

- Discussed moving meeting times for next week / meeting more than once so more people can attend
- So far, Anders and Diego have tested kernels/functions of the form $k = 0.2$ (i.e., constant) with $f = 0$ and $k = \frac{1}{x}$ with $f = x^2 + 2$. Both have code that is running mostly as expected but with some weird "spikes" in certain places.
- We reviewed all ideas currently on the overleaf document

TODO:

- Keep experimenting with different kernels/functions
- Try to find source of spikes in code/methodology
- Write up generalized setup for our problem

PR: Start by providing more backgroun/notation here. State what the interval is $[0, \infty)$ maybe? What is the discretization, what $u_0, u_1 ...$ represent, what kernel you have chosen etc.

I tried writing an introduction below. Please give any suggestions to improve/correct/make this approach more readable. -AB.

3. Introduction

In this paper, we will discuss what are the kernel functions and their properties, and then prove why it can estimate the derivative (in 1 dimensional). Finally, show how well they approximate the derivative in MATLAB.

In the second part, we try to solve DEQs by nonlocal derivatives, which will translate into a solved equation problem in the end. Which may have a large application in solving DEQs with unknowable functions. However, in realistic experiments, the kernel functions are not really stable, which means that for different differential equations, different kernel functions have different errors. So finding the best kernel function is what we should do now. But this is a really tough way to approximate primitive functions because the only way to find an appropriate function is just to try different kernels randomly, it's unrealistic.

During this process, you may notice that finding the kernel function is equivalent to calculating the coefficients in the equations we need to solve directly. So what we did in the third part was calculate the appropriate coefficients directly which skipped the process for searching kernel. We use something similar to the finite difference method. Finally, we gave the approximation results of the primitive function and the convergence rate.$-Zhekai$

3.1. **Discretizing the nonlocal differential equation (Anders).** We will work numerically with an evenly spaced mesh X of the real line where points in this mesh are denoted x_i.

Importantly, we define a kernel function $k(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. In most use cases, we can write the kernel as $k(x - y)$, however, we may also consider studying kernels of the form $k(x, x - y)$. In this document, we work with a discretized kernel evaluated on a subset of our mesh X. We denote the discretized kernel as a finite array of values $(k_{-h}, \dots k_h)$ with $k_i = k(x_{a+i} - x_a)\Delta x$ and where h is called the horizon of our kernel.

Let a function $u : \mathbb{R} \to \mathbb{R}$ be discretized over a bounded interval as follows: $(u_{-2h+1}, \dots u_N)$ with $u_i = u(x_i)$ and where we may choose $N \geq 1$ as desired. With this in mind, we can now evaluate a discretized nonlocal derivative of u as seen below:

$$\mathcal{D}_k u(x_n) = \int need2write = \sum_{i=-h}^{h} k_i u_{n+i} \tag{1}$$

Notice that the discretized classical derivative is a special case of the discretized nonlocal derivative when $h = 1$ and with the kernel $k = (0, -\frac{1}{h}, \frac{1}{h})$ (I think). A nonlocal initial value problem is given as

$$\begin{cases} \mathcal{D}_k u = f(u, x) \\ \text{Initial Conditions} \end{cases}$$

Applying the discrete definition of the nonlocal derivative to the nonlocal initial value problem, we get an object which can be evaluated numerically.

Definition 3.1. Let $u_{-2h+1}, \dots u_0$ be given as initial conditions. A discrete nonlocal initial value problem (DNIVP) is given as

$$\begin{cases} \sum_{i=-h}^{h} k_i u_{n+i} = f(u_n, x_n) \quad \forall n : (1-h) \leq n \leq (N-h) \\ u_{-2h+1}, \dots u_0 \end{cases} \tag{2}$$

Notice this system of equations can be rewritten and solved as a recursive relation

$$u_i = \frac{-k_{i-1}u_{i-1} - k_{i-2}u_{i-2} \cdots + f(u_{i-h}, x_{i-h}) - k_{i-h}u_{i-h} \cdots - k_{i-2h}u_{i-2h}}{k_i} \tag{3}$$

where $i \in 1, 2, \dots N$ and where we can choose N arbitrarily large.

Theorem 3.2 (Hadamard Well Posedness of DNIVP)**.** *Using induction and* (3) *we can probably show this.* ***Note:*** *One thing needed for Hadamard well posedness of the DNIVP is that the solutions behavior changes continuously with initial conditions. My guess is that the reccurence relation may have this property over a bounded interval of n's (i.e. (I think) given N, for every $\epsilon > 0$ there exists δ such that shifting initial conditions by δ means $|u_n - u'_n| < \epsilon$ for every $n \leq N$)?**

We compare the rate of convergence of the nonlocal solution with convergence of the discrete classical solution to the analytic solution of an ODE. Below are some computational findings to this problem of convergence.

For the following findings/observations, we work with the "step" kernel as given below:

$$\mathcal{K}_n(y) = \frac{1}{\delta^2}\chi_{[0,\delta]} - \frac{1}{\delta^2}\chi_{[-\delta,0]} \tag{4}$$

where $\delta = h\Delta x$ is called the horizon of our kernel. This makes it so either $k_i = \frac{1}{\Delta x h^2}$ or $k_i = -\frac{1}{\Delta x h^2}$. Here is an example of an approximation using

Numerical Observations

- For u to converge at every point to the classical solution on a domain, not only must $\delta \to 0$, not only must our number of neighbors $h \to \infty$, but something along the lines of $\Delta x h^2 \to 0$ should hold. I tested this, and certain points will not converge if we merely have $\Delta x h \to 0$ PR: This is very interesting and a type of asymptotic compatibility result, similar to Du&Tian - 2013. We should try to see if the power in h can be sub-quadratic, and also if we can prove this result theoretically, basically Theorem 3.4 below, but I do wonder why we need h quadratic and we should try to get rid of some of the assumptions on f.
- These "certain" points I speak of are the discrete jumps away from the classical solution present in the numerical solution to the DNIVP w/ "step" kernel. (I can provide image if wanted) PR: Yes, please provide the image, it sounds very interesting!

Lemma 3.3. *If our initial conditions $u_{-2h+1} \dots u_0$ approach a differentiable function as $\Delta x \to 0$, then for every i and j in our domain where $|i-j| \leq 2h$, we have that $\frac{u_i - u_j}{\Delta x h}$ is finite in the limit as $\Delta x h \to 0$.*

Proof. We derive our initial conditions $u_{-2h+1} \dots u_0$ from the unique solution to the corresponding IVP $\frac{dw}{dt} = f(w,t)$ and $w(t_0) = w_0$ where $w(t_i) = u_i$.

We proceed by strong induction. Given i, *WLOG* we assume $i - 2h + 1 \leq j < i$, that we have $\frac{u_i - u_j}{\Delta x h}$ is finite.

Base case: set $i = 1$, with our initial conditions and applying (3), we get for $-2h+1 \leq j < 1$ that

$$\frac{u_1 - u_j}{\Delta x h} = \frac{-k_0 u_0 - k_{-1}u_{-1} \cdots + -k_{1-h}u_{1-h} + f(u_{1-h}, t_{1-h}) \cdots - k_{1-2h}u_{1-2h}}{k_1 \Delta x h} - \frac{u_j}{\Delta x h}$$

Since we are using the kernel in (4), this becomes

$$\frac{u_1 - u_j}{\Delta x h} = \frac{-u_j - u_0 - u_{-1} \cdots - 0 + f(u_{1-h}, t_{1-h})\Delta x^2 h^2 \cdots + u_{1-2h}}{\Delta x h}$$
$$\leq M + f(u_{1-h}, t_{1-h})\Delta x h$$

where we see that $M = sup\{\frac{(-u_j+u_{-h})}{\Delta x h}, \frac{(-u_0+u_{-h-1})}{\Delta x h} \dots \frac{(-u_{-h+2}+u_{-2h+1})}{\Delta x h}\}$ is finite since the initial conditions approach a differentiable function. Since $\Delta x h \to 0$ and f is bounded, the whole expression is finite.

Induction step: we work with $i+1$,

$$\frac{u_{i+1} - u_j}{\Delta x h} = \frac{-u_j - u_{i+1} - u_i \cdots - 0 + f(u_{i+1-h}, x_{i+1-h})\Delta x^2 h^2 \cdots + u_{i+1-2h}}{\Delta x h}$$

By the same argument above, we find that this is finite. □

The below theorem is motivated by computational findings, however it has yet to be correctly proven. Recall that Δx is our mesh size and h is the number of neighbors we are including on each side of our kernel.

Theorem 3.4 (Convergence of DNIVP solution to Euler's method on classical solution)**.** *Let $\mathcal{D}_k u = f(u,t)$ be a DNIVP where k is the "step" kernel (4) and let $w' = f(w,t)$. We work with numerical solutions u_i, w_i to these problems on our mesh T and where the w_i are found using eulers method.*

- *Assume $f(y(t),t)$ is bounded, locally Lipschitz in y, and continuous in t.*
- *Assume $\Delta x h^2 \to 0$ where Δx is our mesh size and h is the number of neighboring points to either side used in our kernel. Note that this implies our horizon $\delta = \Delta x h \to 0$.*
- *Assume we derive the initial conditions $u_{-2h+1} \dots u_0$ to the DNIVP from w evaluated at mesh points such that $u_0 = w_0$*

Then each u_i converges to w_i as $\Delta x h^2 \to 0$, (and $h \to \infty$–currently unused assumption but numerical results imply that it is sometimes necessary. This is a major problem).

Proof. We want to show that $|u_i - w_i|$ approaches 0 as $\Delta x h^2 \to 0$ (and $h \to \infty$). We proceed by strong(?) induction on u. We assume that for all $1 \leq n \leq 2h$ we have $u_{i-n} \to w_{i-n}$ as $\Delta x h^2 \to 0$.

We have that

$$|w_{2h+1} - u_{2h+1}| = |f(w_{2h}, t_{2h})\Delta x - \frac{f(u_h, t_h)}{k_{2h+1}} + w_{2h} + u_{2h} + \cdots - u_2 - u_1|$$

Rearranging terms we have

$$\leq |f(w_{2h}, t_{2h})\Delta x| + |(w_{2h} - u_h) + (u_{2h} - u_{h-1}) \cdots + (u_{h+2} - u_1) - \frac{f(u_h, t_h)}{k_{2h+1}}|$$
$$\leq |f(w_{2h}, t_{2h})\Delta x| + |M\Delta x h^2 - f(u_h, t_h)\Delta x h^2|$$
$$= |f(w_{2h}, t_{2h})\Delta x| + |\Delta x h^2||M - f(u_h, t_h)|$$

where $M = sup\{\frac{(w_{2h}-u_h)}{\Delta x h}, \frac{(u_{2h}-u_{h-1})}{\Delta x h}, \dots \frac{(u_{h+2}-u_1)}{\Delta x h}\}$. PR: I wonder if this is the optimal arrangement of terms to use the triangle inequality. I think that we should be able to do it without using that f is bounded.

For this to approach 0, knowing that $\Delta x h^2 \to 0$, we must show M is finite. In the case that $M = \frac{(u_{2i}-u_{2i-(h+1)})}{\Delta x h}$, since our initial conditions are derived from a differentiable function $w(t)$, by lemma 3.3 we see this is true. In the case that $M = \frac{(w_{2h}-u_h)}{\Delta x h}$, by backtracking through Euler's method we know that

$$w_{2h} = w_0 + \Delta x(f(w_0, t_0) \cdots + f(w_{2h-1}, t_{2h-1}))$$

$$\leq u_0 + \Delta x h L$$

where $L = sup\{f(w_0, t_0) \cdots + f(w_{2h-1}, t_{2h-1})\}$. So we now have

$$\frac{w_{2h} - u_h}{\Delta x h} \leq \frac{u_0 - u_h + \Delta x L'}{\Delta x h}$$

which does go to zero again by lemma 3.3.

For clarity, expanding my notation, we have:

$$\frac{w_{2h} - u_h}{h\Delta x} = \frac{w(2h\Delta x + t_0) - u(h\Delta x + t_0)}{h\Delta x}$$

□

4. -Zhekai Properties of the kernel function

4.1. **What kind of function can be a kernel to approximate the derivative?** Before we solve the DNIVP equations, let's first look at the kernel function and why it can be used instead of the derivative in differential equations, and visualize the effects of them.

We want a good function $\mathcal{K}$ where $\int_{-\infty}^{+\infty} \mathcal{K}(y)dy = 0$. Let $\delta = 1/n, n \in N$ Suppose we have the sequence $\{\mathcal{K}_i\}_{i\in\mathbb{N}}$ and $f(x)$ is a smooth function. Do a Taylor expansion of function f (at least 3 times differentiable):

$$f(x+y) = f(x) + f'(x)y + \frac{f''(x)}{2!}y^2 + \frac{f'''(\xi_x)}{3!}y^3$$

$$\begin{aligned}
\mathcal{D}_{\mathcal{K}_n} f(x) &= \int_{-\infty}^{+\infty} \left[f(x) + f'(x)y + \frac{f''(x)}{2!}y^2 + \frac{f'''(\xi_x)}{3!}y^3 - f(x) \right] \mathcal{K}_n(y)\mathrm{d}y \\
&= \int_{-\infty}^{+\infty} f'(x) y k_n(y)\mathrm{d}y + \int_{-\infty}^{+\infty} \frac{f''(x)}{2!}y^2 k_n(y)\mathrm{d}y + \int_{-\infty}^{+\infty} \frac{f'''(\xi_n)}{3!}y^3 k_n(y)\mathrm{d}y \\
&= f'(x)\int_{-\infty}^{+\infty} y k_n(y)\mathrm{d}y + \frac{f'''(\xi_n)}{3!}\int_{-\infty}^{+\infty} y^3 k_n(y)\mathrm{d}y \\
&\approx f'(x)
\end{aligned}$$

If and only if:

- $\lim_{n\to\infty} \int_{-\infty}^{+\infty} y\mathcal{K}_n(y)\mathrm{d}y = 1$
- $\lim_{n\to\infty} \int_{-\infty}^{+\infty} y^3\mathcal{K}_n(y)\mathrm{d}y = 0$
- $\int_{0}^{+\infty} y^2\mathcal{K}_n(y)\mathrm{d}y < +\infty$

For example, about step function:

$$\mathcal{K}(y) = C\chi_{[0,h]} - C\chi_{[-h,0]}$$

Where χ is the indicator function. To satisfy the first property:

$$\int_{-\infty}^{+\infty} y\left(C\chi_{[0,h]} - C\chi_{[-h,0]}\right)\mathrm{d}y = Ch^2 = 1$$

So:

$$C = \frac{1}{h^2}$$

So the step kernel can be written as:

$$\mathcal{K}_n(y) = \frac{1}{h^2}\chi_{[0,h]} - \frac{1}{h^2}\chi_{[-h,0]}$$

Where $h = \frac{1}{n}$.

By using this kernel function, to approximate the derivative of $cos(x)$[FIGURE 1]: Now let us try a new kernel:

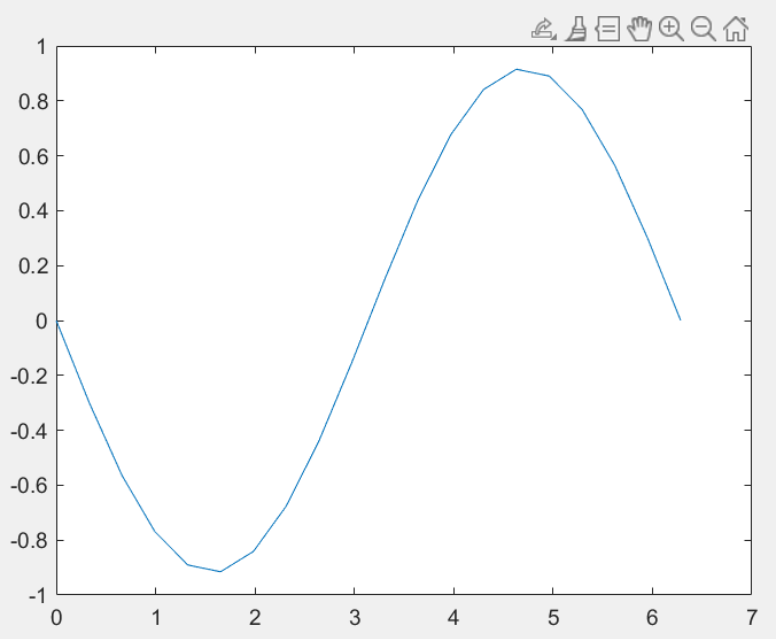

FIGURE 1. Approximate the derivative of $cos(x)$

$$\mathcal{K}(y) = \frac{1}{y^{0.5}}\chi_{(0,\alpha]} - \frac{1}{|y|^{0.5}}\chi_{[-\alpha,0)}$$

Where $\alpha = (0.75)^{\frac{2}{3}}$. (To satisfy first property of the kernel function).

We can compare this kernel function with Step kernel function $\mathcal{K}_n(y) = \frac{1}{h^2}\chi_{[0,h]} - \frac{1}{h^2}\chi_{[-h,0]}$ (where $n = 2$) [FIGURE 2]:

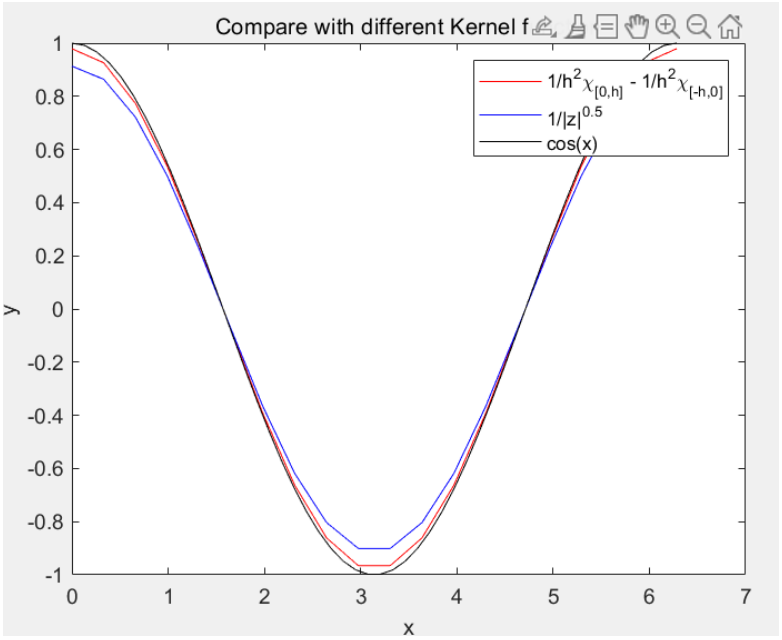

FIGURE 2. Comparison and contrast

5. OLD

Given a DNIVP as follows (section needs to be updated)

$$\mathcal{D}_k u = f(u, x)$$

and given a kernel $k = (a_{-2}, a_{-1}, ...a_{11})$ the kernel would have to be evaluated at the mesh point. I suggest to start with a kernel and then define the vector/array of values k_i as the evaluations of the continuous kernel at the mesh points with a horizon $\delta = 2h$ where h is the mesh size. PR: where h is the mesh size

From this we can derive have a system of equations with 14 unknowns and 10 equations:

$$\begin{cases} \mathcal{D}_k u_0 = f(u_0, x_0) & = a_{-2}u_{-2} + a_{-1}u_{-1} + a_0 u_0 + a_1 u_1 + a_2 u_2 \\ \mathcal{D}_k u_1 = f(u_1, x_1) & = a_{-1}u_{-1} + a_0 u_0 + a_1 u_1 + a_2 u_2 + a_3 u_3 \\ \mathcal{D}_k u_2 = f(u_2, x_2) & = a_0 u_0 + a_1 u_1 + a_2 u_2 + a_3 u_3 + a_4 u_4 \\ & \vdots \\ \mathcal{D}_k u_9 = f(u_9, x_9) & = a_7 u_7 + a_8 u_8 + a_9 u_9 + a_{10} u_{10} + a_{11} u_{11} \end{cases}$$

where $u(x_n) = u_n$. With some manipulation, our system becomes:

$$\begin{cases} -a_{-2}u_{-2} - a_{-1}u_{-1} - a_0 u_0 - a_1 u_1 & = a_2 u_2 - f(u_0, x_0) \\ -a_{-1}u_{-1} - a_0 u_0 - a_1 u_1 & = a_2 u_2 + a_3 u_3 - f(u_1, x_1) \\ -a_0 u_0 - a_1 u_1 & = a_2 u_2 + a_3 u_3 + a_4 u_4 - f(u_2, x_2) \\ -a_1 u_1 & = a_2 u_2 + a_3 u_3 + a_4 u_4 + a_5 u_5 - f(u_3, x_3) \\ 0 & = a_2 u_2 + a_3 u_3 + a_4 u_4 + a_5 u_5 + a_6 u_6 - f(u_4, x_4) \\ & \vdots \\ 0 & = a_7 u_7 + a_8 u_8 + a_9 u_9 + a_{10} u_{10} + a_{11} u_{11} - f(u_9, x_9) \end{cases}$$

for which working on a case by case basis with $f(u_n, x_n)$ can be written as a matrix system taking u_{-2}, u_{-1}, u_0, u_1 to be given as initial conditions. This gives us a system with 10 unknowns and 10 equations, meaning there is a unique solution $u = (u_0, u_1, ...u_9)$ to this system (I'd guess there can't be degenerate cases here w/ no sol but correct me if I'm wrong).

Something to note is that this kernel $k = (a_{-2}, a_{-1}, ...a_{11})$ is completely x dependent and not dependent on $|x - y|$ as the kernels we have been working with previously have been. But we can do whatever with the kernels. Also, (to be written up nicely), we can generalize the above process for any horizon and to solve for any subset of the domain of our solution u.

PR: Very good start here. I would suggest that some of you start working on the implementation in Matlab, while others start to generalize this setup and see what would be needed for solvability. Then start playing with the horizon, see if you can capture what the numerical solution would do, how far apart would be from the theoretical solution etc.

6. -Zhekai Solve DEQs by the kernel function

In this section, we want to solve ODEs by solving equations we present in [section 5], the equations can be generalized like:

$$\mathcal{D}u(x) = f(x) = \sum_{j=-m}^{m} a_j u(x + jh)$$

Where $m = 2$ here, and a_i depends on the kernel function and the distance of each step. The algorithm is like that 1:

Algorithm 1 The algorithm to solve ODEs.

Require: ODE: $x' = cos(x)$, Initial conditions u_1, u_2, u_{N-1}, u_N where N is the number of mesh points and the kernel function: $\mathcal{K}(y) = \frac{1}{y^{0.5}}\chi_{(0,\alpha]} - \frac{1}{|y|^{0.5}}\chi_{[-\alpha,0)}$ Where $\alpha = (0.75)^{\frac{2}{3}}$.

1: Create matrix $A^{(N-4)\times N} = a_{i,j}$
2: Create mesh points $x = linspace(0, 2\pi, N)$
3: Create vector $b = cos(x)$
4: **for** i in 3 to $N - 3$ **do**
5: $\quad a_{i,N-2} = \mathcal{K}(|a_i - a_{N-2}|)$
6: $\quad a_{i,N-1} = \mathcal{K}(|a_i - a_{N-1}|)$
7: $\quad a_{i,N} = 0$
8: $\quad a_{i,N+2} = \mathcal{K}(|a_i - a_{N+2}|)$
9: $\quad a_{i,N+1} = \mathcal{K}(|a_i - a_{N+1}|)$
10: **end for**
11: Solve $Au = b$
12: **return:** plot the figure (x, u).

At first, we use 10 equations to solve 14 unknown points and give the initial conditions u_1, u_2, u_{13}, u_{14}, the final result is below 3:

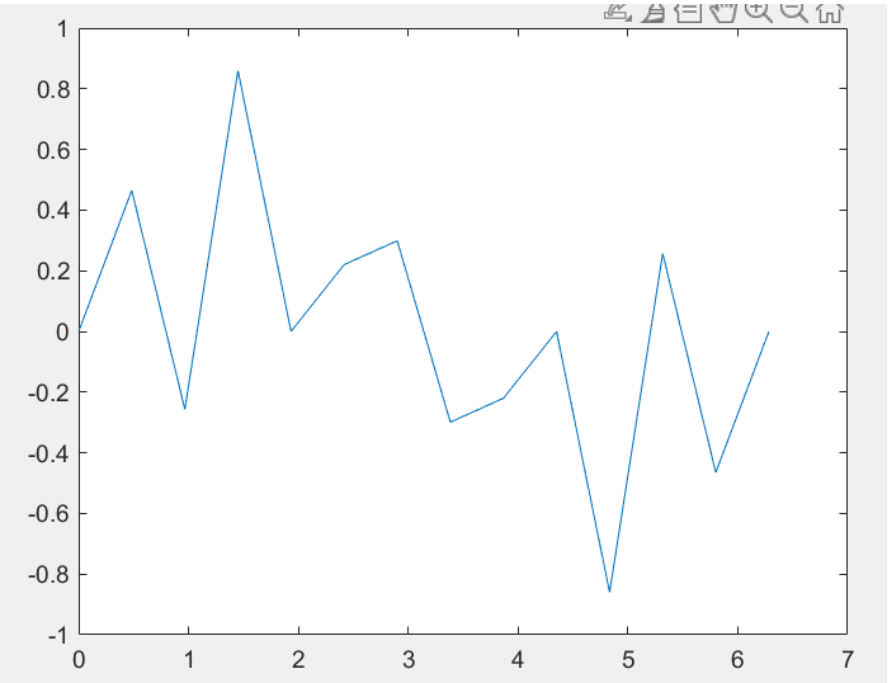

FIGURE 3. $N = 14$ the true solution is $sin(x)$

The result is not really precise. Then, we use 26 equations to solve 30 unknown points and give the initial conditions u_1, u_2, u_{29}, u_{30}, the final result is below 4:

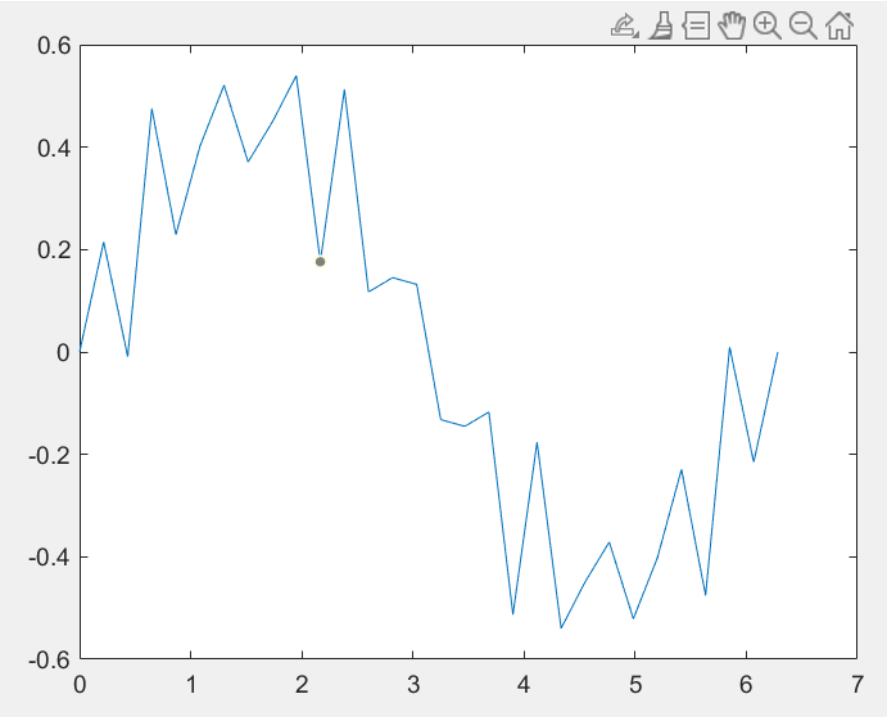

FIGURE 4. $N = 30$ the true solution is $sin(x)$

The result is not really precise. Finally, we use 96 equations to solve 100 unknown points and give the initial conditions $u_1, u_2, u_{99}, u_{100}$, the final result is below 5:

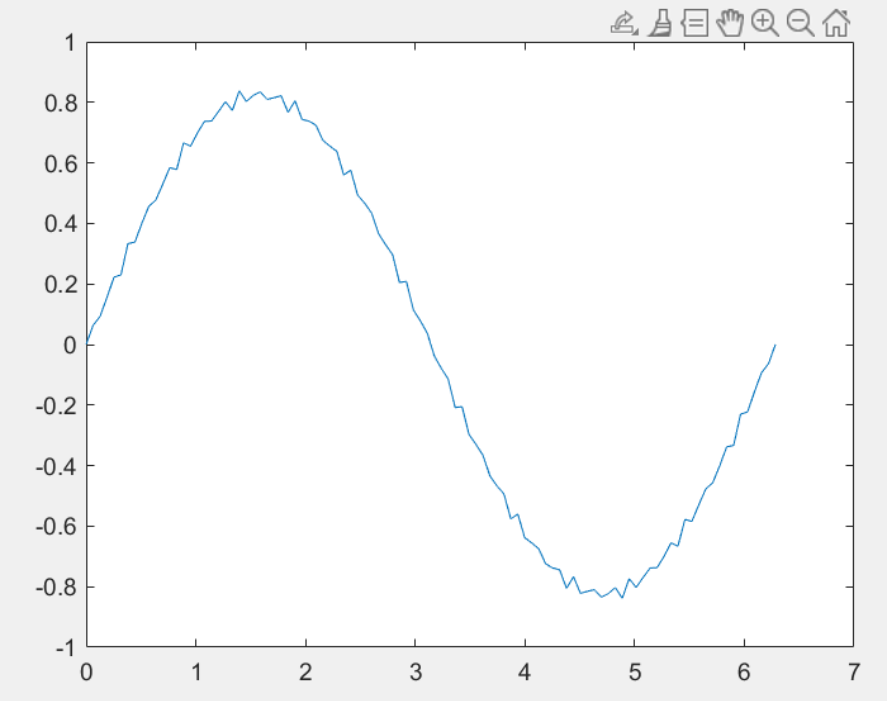

FIGURE 5. $N = 100$ the true solution is $sin(x)$

However, if we look more closely, we will find the peak of the solution function cannot reach 1. That is a terrible thing. In order to solve that, we have tried lots of methods. For instance, we increase N and each side's points number but all of them have failed.

Finally, I think the big error comes from the kernel function. So I begin to tune the parameter of the kernel function. Coincidently, I find nonlocal-solutions perform really will if we use the kernel: $\mathcal{K}(y) = \frac{1}{y^{0.4}}\chi_{(0,\alpha]} - \frac{1}{|y|^{0.4}}\chi_{[-\alpha,0)}$[Figure 6]:

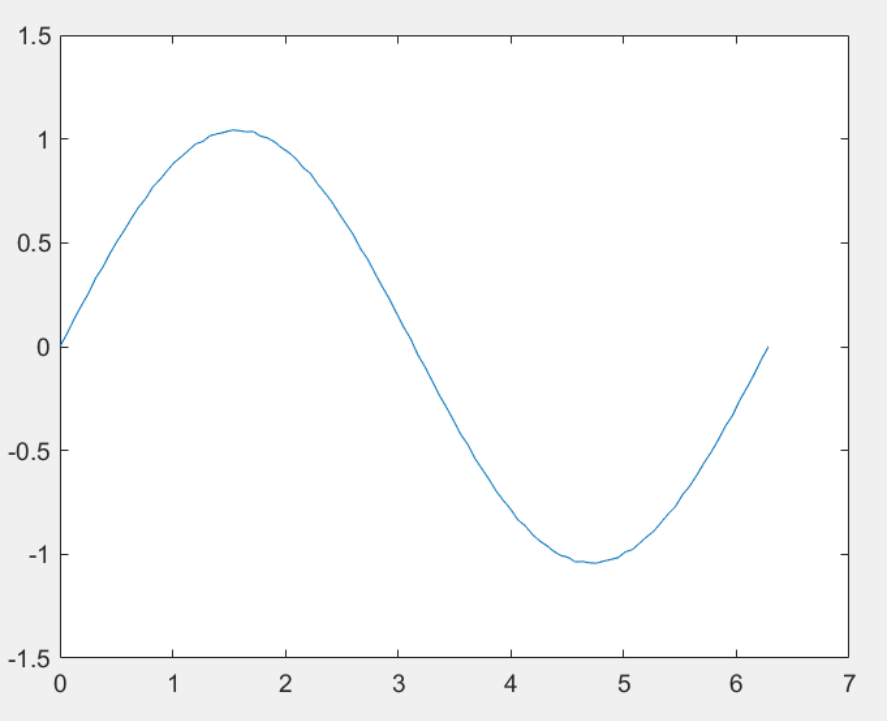

FIGURE 6. $N = 100$ the true solution is $sin(x)$

The experiments show that we can truly solve the ODEs by this method. As we discussed in the abstract, maybe we can use the nonlocal derivative method to solve some unknowable functions in black box optimizing situations. What's more from this observation, we know that the kernel function is extremely important. We can also notice that the coefficients $\{a_i\}$ of the equations depend on the kernel function. So finding a good kernel is equivalent to finding suitable coefficients. Now let us consider the finite difference method, to find proper coefficients.

7. Jennifer - nonlocal derivative vs finite difference method

The nonlocal derivative can be expressed as:

$$Du(x) = \sum_{j=-m}^{m} a_j u(x + jh) = f(x),$$

where a_j are the coefficients and h is the step size.

This formulation is essentially the same idea as the finite difference method. A more precise way to write this is:

$$Du(x) = \frac{1}{h} \sum_{j=-m}^{m} a_j u(x + jh).$$

Using a Taylor expansion, we can express $u(x + jh)$ as:

$$u(x + jh) = u(x) + jhu'(x) + \frac{(jh)^2}{2!}u''(x) + \frac{(jh)^3}{3!}u'''(x) + \cdots .$$

Substituting this Taylor series into the nonlocal derivative expression, we get:

$$
\begin{aligned}
Du(x) &= \frac{1}{h}\sum_{j=-m}^{m} a_j \left(u(x) + jhu'(x) + \frac{j^2h^2}{2}u''(x) + \frac{j^3h^3}{6}u'''(x) + \cdots \right) \\
&= \frac{1}{h}\left(\sum_{j=-m}^{m} a_j u(x) + \sum_{j=-m}^{m} a_j jhu'(x) + \sum_{j=-m}^{m} a_j \frac{j^2h^2}{2}u''(x) + \sum_{j=-m}^{m} a_j \frac{j^3h^3}{6}u'''(x) + \cdots \right) \\
&= \frac{1}{h}\left(\left(\sum_{j=-m}^{m} a_j \right) u(x) + h\left(\sum_{j=-m}^{m} ja_j \right) u'(x) + \frac{h^2}{2}\left(\sum_{j=-m}^{m} j^2a_j \right) u''(x) + \frac{h^3}{6}\left(\sum_{j=-m}^{m} j^3a_j \right) u'''(x) + \right.
\end{aligned}
$$

To ensure that the result approximates $u'(x)$ as closely as possible, we impose the following conditions:

$$
\begin{gathered}
\sum_{j=-m}^{m} a_j = 0, \\
\sum_{j=-m}^{m} ja_j = 1, \\
\sum_{j=-m}^{m} j^2a_j = 0, \\
\sum_{j=-m}^{m} j^3a_j = 0, \\
\vdots \\
\sum_{j=-m}^{m} j^{2m}a_j = 0.
\end{gathered}
$$

This leads to a system of equations, which can be written in matrix form as follows:

$$
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 & 1 \\
-m & -(m-1) & -(m-2) & \cdots & (m-1) & m \\
m^2 & (m-1)^2 & (m-2)^2 & \cdots & (m-1)^2 & m^2 \\
-m^3 & -(m-1)^3 & -(m-2)^3 & \cdots & (m-1)^3 & m^3 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
m^{2m} & (m-1)^{2m} & (m-2)^{2m} & \cdots & (m-1)^{2m} & m^{2m}
\end{pmatrix}
\begin{pmatrix}
a_{-m} \\ a_{-(m-1)} \\ a_{-(m-2)} \\ \vdots \\ a_{m-1} \\ a_m
\end{pmatrix}
=
\begin{pmatrix}
0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0
\end{pmatrix}
$$

This matrix is solvable because it is a square Vandermonde matrix. The determinant of a square Vandermonde matrix is called a Vandermonde determinant or Vandermonde polynomial. Its value is:

$$
\det(V) = \prod_{0 \le i < j \le n} (x_j - x_i).
$$

This is non-zero if and only if all x_i are distinct (no two are equal), making the Vandermonde matrix invertible.

Given that $-m, -(m-1), \ldots, m-1, m$ are distinct by construction, the primary condition for solvability is inherently satisfied. Therefore, the matrix is invertible, and the system has a unique solution.

The order of accuracy for this method in theory is $O(h^{2m})$.

(there might be a general formula representing a_j?

Algorithm 2 Compute Nonlocal Solution

Require: $extended\_x$, x, f, u, k, h
Ensure: $u\_interior$
$N \leftarrow \text{length}(extended\_x)$
$M \leftarrow \text{length}(x)$ {Number of interior points to solve for}
$n \leftarrow 2k + 1$ {Size of the matrix}
Initialize the matrix A_kernel of size $n \times n$ with zeros
for $i = 1$ to n **do**
  for $j = -k$ to k **do**
    $colIndex \leftarrow j + k + 1$
    $A\_kernel[i, colIndex] \leftarrow (-j)^{(i-1)} \cdot (-1)^{(i+1)}$
  end for
end for
Initialize the vector b_kernel of size n with zeros
$b\_kernel[2] \leftarrow \frac{1}{h}$ {The second element is $\frac{1}{h}$, rest are 0}
Solve for the coefficients a using $a \leftarrow \text{linsolve}(A\_kernel, b\_kernel)$
Initialize the matrix A of size $M \times M$ with zeros
Initialize the vector b of size M with zeros
for $i = 1$ to M **do**
  for $j = -k$ to k **do**
    if $(i + j) > 0$ **and** $(i + j) \leq M$ **then**
      $A[i, i + j] \leftarrow a[j + k + 1]$
    end if
  end for
  $b[i] \leftarrow \cos(x[i])$
end for
Apply boundary conditions:
$b[1] \leftarrow b[1] - a[1] \cdot u[1] - a[2] \cdot u[2]$
$b[2] \leftarrow b[2] - a[1] \cdot u[2]$
$b[M - 1] \leftarrow b[M - 1] - a[\text{end}] \cdot u[\text{end-1}]$
$b[M] \leftarrow b[M] - a[\text{end}] \cdot u[\text{end}] - a[\text{end-1}] \cdot u[\text{end-1}]$
Solve the linear system for the interior points $u\_interior \leftarrow \text{linsolve}(A, b)$

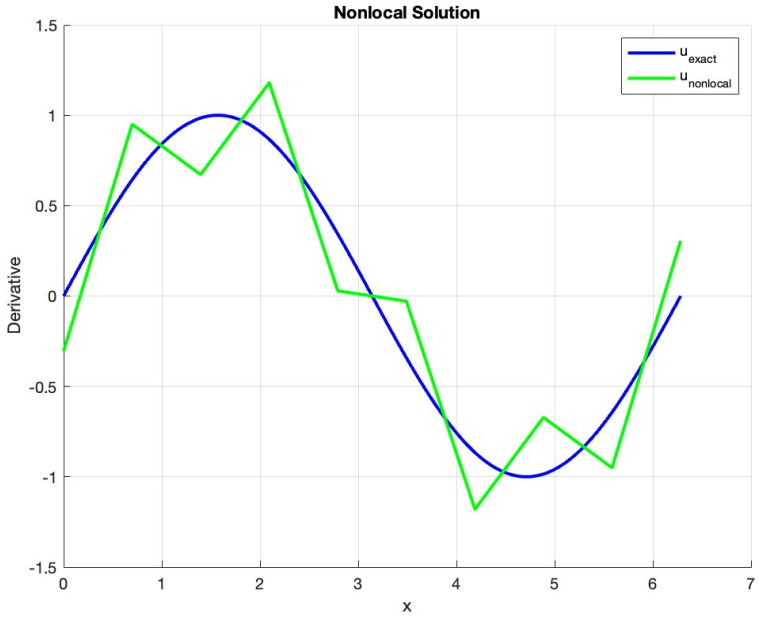

FIGURE 7. Nonlocal Solution for $N = 10, m = 5$

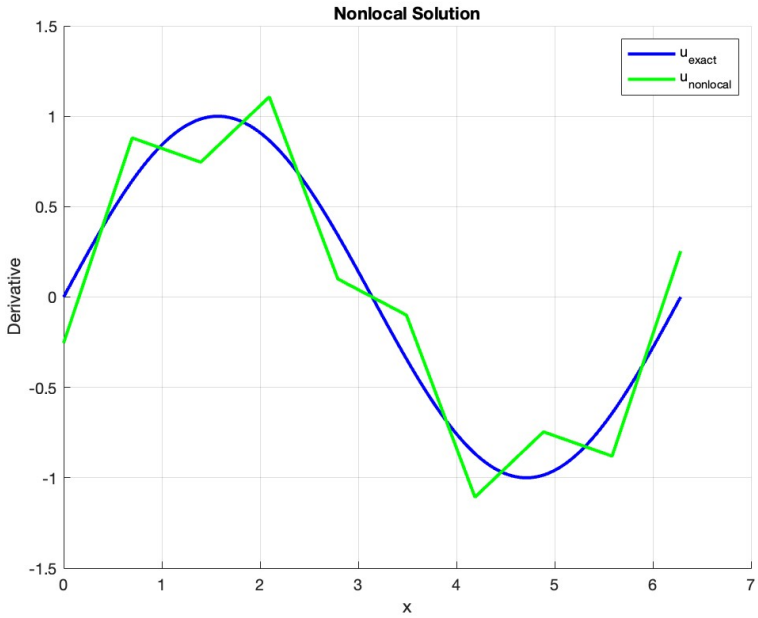

FIGURE 8. Nonlocal Solution for $N = 10, m = 10$

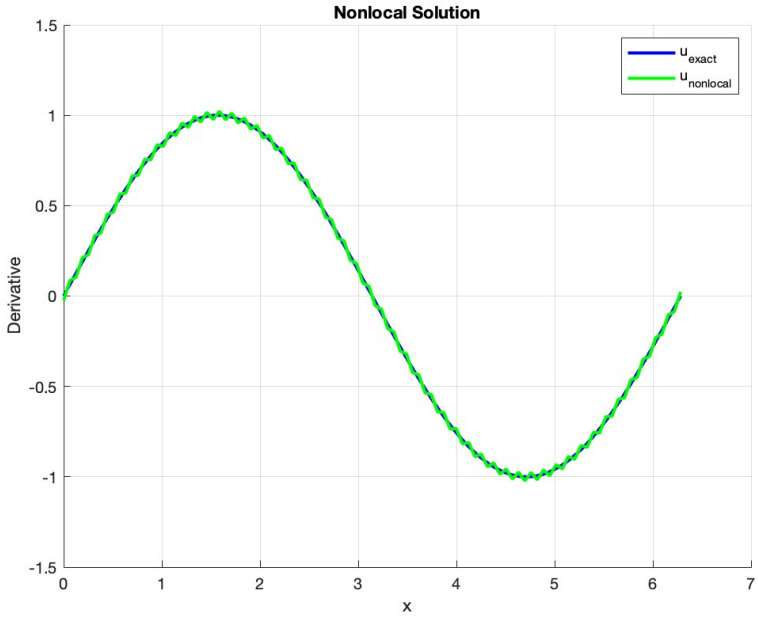

FIGURE 9. Nonlocal Solution for $N = 100, m = 10$

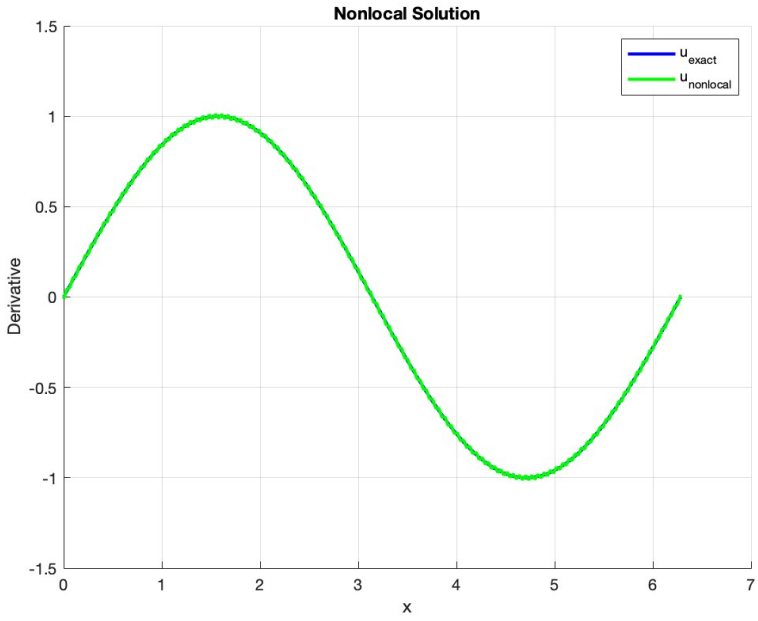

FIGURE 10. Nonlocal Solution for $N = 200, m = 10$

Spacial Discretization N= 5000	
m	L2 Error
1	4.67e-07
2	1.53e-13
3	2.58e-03
4	1.23e-03
5	1.50e-03
6	1.28e-03
7	1.25e-03
8	1.18e-03
9	1.13e-03
10	1.09e-03
11	1.06e-03
12	5.54e-03
13	9.70e-02
>=14	the result does not make sense anymore

It seems that the nonlocal operator

$$Du(x) = \sum_{j=-m}^{m} a_j u(x+jh) = f(x),$$

works best when $m = 2$, which corresponds to a five-point stencil. PR: This is also an observation that was made in different applications, I do wonder why it is not $m = 3$ or another value. Maybe soemthing to be proven theoretically.

7.0.1. *one-side kernel.* This method can also be extended to one side kernel.

Consider, only the left hand side kernel, then

$$\begin{aligned} Du(x) &= \frac{1}{h} \sum_{j=-m}^{0} a_j u(x+jh) \\ &= \frac{1}{h} \left(\left(\sum_{j=-m}^{0} a_j \right) u(x) + h \left(\sum_{j=-m}^{0} j a_j \right) u'(x) + \frac{h^2}{2} \left(\sum_{j=-m}^{0} j^2 a_j \right) u''(x) + \frac{h^3}{6} \left(\sum_{j=-m}^{0} j^3 a_j \right) u'''(x) + \right. \end{aligned}$$

To ensure that the result approximates $u'(x)$ as closely as possible, we impose the following conditions:

$$\sum_{j=-m}^{0} a_j = 0,$$

$$\sum_{j=-m}^{0} j a_j = 1,$$

$$\sum_{j=-m}^{0} j^2 a_j = 0,$$

$$\vdots$$

$$\sum_{j=-m}^{0} j^m a_j = 0.$$

This leads to a system of equations, which can be written in matrix form as follows:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ -m & -(m-1) & -(m-2) & \cdots & (m-1) & m \\ m^2 & (m-1)^2 & (m-2)^2 & \cdots & (m-1)^2 & m^2 \\ -m^3 & -(m-1)^3 & -(m-2)^3 & \cdots & (m-1)^3 & m^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ m^m & (m-1)^m & (m-2)^m & \cdots & (m-1)^m & m^m \end{pmatrix} \begin{pmatrix} a_{-m} \\ a_{-(m-1)} \\ a_{-(m-2)} \\ \vdots \\ a_{-1} \\ a_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

which is also solvable, the order of accuracy in theory is $O(h^m)$

Spacial Discretization N= 5000	
m	L2 Error
1	1.93e-03
2	9.33e-07
3	5.78e-03
4	5.51e-03
5	1.17e-02
6	1.16e-02
7	1.25e-03
>=7	the result does not make sense anymore

7.0.2. *Key differences between nonlocal models and finite difference methods:*

(1) Weights: Finite difference methods use fixed weights derived from Taylor series expansions around a point. Nonlocal models allow for more flexibility by using different weights, often determined by specific kernels. These kernels can assign varying weights at different points to compute different types of rates of change.
(2) Horizon and Symmetry: For finite difference method, the set of points for approximation is usually fixed and symmetric around the point of interest. Nonlocal models can choose different horizons for the kernel, which do not have to be symmetric. This allows for the inclusion of points that are further away or closer to the point of interest, depending on the specific problem and desired accuracy.
(3) The coefficients for finite difference methods are generally computed to be optimal for a broad class of functions, typically all smooth functions. However, it is possible for nonlocal models to compute optimal coefficients for specific classes of functions, such as odd functions or polynomials.

7.0.3. *Generalization - given the kernel.* Consider a more general case where we are given the kernel $k(x, y-x)$, which represents a heterogeneous kernel. This kernel can vary with both the position x and the relative distance $y-x$.

The nonlocal derivative is defined as:

$$Du(x) = \sum_{j=-n}^{m} k(x, jh)u(x+jh),$$

where the kernel does not have to be symmetric.

We aim to determine what kind of differential operator $Du(x)$ best approximates.

Using a Taylor expansion, we can express $u(x+jh)$ as:

$$u(x+jh) = u(x) + jhu'(x) + \frac{(jh)^2}{2!}u''(x) + \frac{(jh)^3}{3!}u'''(x) + \cdots .$$

Substituting this Taylor series into the nonlocal derivative expression, we get:

$$\begin{aligned} Du(x) &= \sum_{j=-n}^{m} k(x,jh)\left(u(x) + jhu'(x) + \frac{j^2h^2}{2}u''(x) + \frac{j^3h^3}{6}u'''(x) + \cdots\right) \\ &= \sum_{j=-n}^{m} k(x,jh)u(x) + \sum_{j=-n}^{m} k(x,jh)jhu'(x) + \sum_{j=-n}^{m} k(x,jh)\frac{j^2h^2}{2}u''(x) + \sum_{j=-n}^{m} k(x,jh)\frac{j^3h^3}{6}u'''(x) + \cdots \\ &= \left(\sum_{j=-n}^{m} k(x,jh)\right)u(x) + h\left(\sum_{j=-n}^{m} jk(x,jh)\right)u'(x) + \frac{h^2}{2}\left(\sum_{j=-n}^{m} j^2k(x,jh)\right)u''(x) + \frac{h^3}{6}\left(\sum_{j=-n}^{m} j^3k \right. \end{aligned}$$

To identify the coefficients a_i in the best approximation of the differential operator $a_0(x)u(x) + a_1(x)u'(x) + a_2(x)u''(x) + \cdots$, we set:

$$a_0(x) = \sum_{j=-n}^{m} k(x,jh),$$

$$a_1(x) = h\left(\sum_{j=-n}^{m} jk(x,jh)\right),$$

$$a_2(x) = \frac{h^2}{2}\left(\sum_{j=-n}^{m} j^2k(x,jh)\right),$$

$$\vdots$$

To approximate this operator using the nonlocal derivative with a heterogeneous kernel, we design the kernel $k(x,jh)$ to match the terms in the differential operator. The kernel $k(x,jh)$ should ensure that the coefficients $a_0(x), a_1(x), \ldots$ are correctly approximated.

For example, if we want $D_1u(x) = a_0(x)u(x) + a_1(x)u'(x)$, we must set:

$$\sum_{j=-n}^{m} k(x,jh) = a_0(x),$$

$$\sum_{j=-n}^{m} jk(x,jh) = \frac{a_1(x)}{h},$$

$$\sum_{j=-n}^{m} j^2k(x,jh) = 0,$$

$$\sum_{j=-n}^{m} j^3k(x,jh) = 0,$$

$$\vdots$$

8. Jennifer - Notes from 7.29

(IVP)

$$\begin{cases} D_k u(x) = f(x, u(x)), & x \in [-\delta, \infty) \\ u(x) = u_0(x), & x \in [-2\delta, 0] \end{cases}$$

Write the IE: (We assumed so far compact support symmetrically distributed on $[-\delta, \delta]$).

$$\int_{x-\delta}^{x+\delta} [u(y) - u(x)]k(y-x)\,dy = f(x, u(x))$$

Evaluate at $x - \delta$:

$$\int_{x-2\delta}^{x} [u(y) - u(x-\delta)]k(y-x+\delta)\,dy = f(x-\delta, u(x-\delta))$$

$$\Leftrightarrow \int_{x-2\delta}^{0} \underbrace{u(y)}_{u_0(y)} k(y-x+\delta)\,dy + \int_0^x u(y)k(y-x+\delta)\,dy - \int_{x-2\delta}^{x} \underbrace{u(x-\delta)}_{=u_0(x-\delta)} k(y-x+\delta)\,dy = f(x-\delta, u(x-\delta))$$

$$\Leftrightarrow \int_0^x u(y)k(y-x+\delta)\,dy = \underbrace{-\int_{x-2\delta}^{0} u_0(y)k(y-x+\delta)\,dy + \int_{x-2\delta}^{x} u_0(x-\delta)k(y-x+\delta)\,dy + f(x-\delta, u(x-}_{\text{known function, denote by } \tilde{f}(x-\delta, u(x-\delta))}$$

Hence, we need to solve for u which satisfies:

$$\int_0^x u(y)k(y-x+\delta)\,dy = \tilde{f}(x-\delta, u(x-\delta)), \quad \forall x \in [0, \delta]$$

Differentiate w.r.t. x both sides (also assume that $\tilde{f}$ only depends on $x - \delta$, for now, this means that u only depends on x). We obtain:

$$u(x)k(\delta) = \tilde{f}'(x-\delta) \Rightarrow u(x) = \frac{1}{k(\delta)}\tilde{f}'(x-\delta), \quad x \in [0, \delta]$$

There are details/questions that arise from the above argument:

(1) Check all details, in particular the continuation $[\delta, 2\delta]$ etc. Will need differentiability for the constructed solution u.
Evaluate at $x - 2\delta$:

$$\int_{x-3\delta}^{x-\delta} [u(y) - u(x-2\delta)]k(y-x+2\delta)\,dy = f(x-2\delta, u(x-2\delta))$$

$$\Leftrightarrow \int_{x-3\delta}^{0} \underbrace{u(y)}_{u_0(y)} k(y-x+2\delta)\,dy + \int_0^{x-\delta} u(y)k(y-x+2\delta)\,dy - \int_{x-3\delta}^{x-\delta} \underbrace{u(x-2\delta)}_{=u_0(x-2\delta)} k(y-x+2\delta)\,dy = f(x-2\delta, u(x-2\delta))$$

$$\Leftrightarrow \int_0^{x-\delta} u(y)k(y-x+2\delta)\,dy = \underbrace{-\int_{x-3\delta}^{0} u_0(y)k(y-x+2\delta)\,dy + \int_{x-3\delta}^{x-\delta} u_0(x-2\delta)k(y-x+2\delta)\,dy + f(x-2}_{\text{known function, denote by } \tilde{f}(x-2\delta, u(x-2\delta))}$$

Hence, we need to solve for u which satisfies:

$$\int_0^{x-\delta} u(y)k(y-x+2\delta)\,dy = \tilde{f}(x-2\delta, u(x-2\delta)), \quad \forall x \in [\delta, 2\delta]$$

Differentiate w.r.t. x both sides (also assume that $\tilde{f}$ only depends on $x-2\delta$, for now, this means that u only depends on x). We obtain:

$$u(x-\delta)k(\delta) = \tilde{f}'(x-2\delta) \Rightarrow u(x-\delta) = \frac{1}{k(\delta)}\tilde{f}'(x-2\delta) \Rightarrow u(x) = \frac{1}{k(\delta)}\tilde{f}'(x-\delta), \quad x \in [\delta, 2\delta]$$

Generalize this:

For general intervals $[n\delta,(n+1)\delta]$:

$$\int_0^{x-n\delta} u(y)k(y-x+(n+1)\delta)\,dy = \tilde{f}(x-(n+1)\delta, u(x-(n+1)\delta)), \quad \forall x\in[n\delta,(n+1)\delta]$$

Differentiate w.r.t. x both sides:

$$u(x-n\delta)k(\delta) = \tilde{f}'(x-(n+1)\delta) \Rightarrow u(x-n\delta) = \frac{1}{k(\delta)}\tilde{f}'(x-(n+1)\delta) \Rightarrow u(x) = \frac{1}{k(\delta)}\tilde{f}'(x-\delta), \quad x\in[n\delta,(n+1)\delta$$

(2) Can we extend above to kernels that are not symmetric on $[-\delta,\delta]$? It seems that we need support on both sides of x, but I am not sure. We should be able to do $k(y-x)$.

Assume that the kernel has support on $[-a,b]\subseteq[-\delta,\delta]$.

$$\int_{x-a}^{x+b}[u(y)-u(x)]k(y-x)\,dy = f(x,u(x)), x\in[0,\delta]$$

Evaluate at $x-\delta$:

$$\int_{x-a-\delta}^{x+b-\delta}[u(y)-u(x-\delta)]k(y-x+\delta)\,dy = f(x-\delta,u(x-\delta))$$

$$\Leftrightarrow \int_{x-a-\delta}^{0}\underbrace{u(y)}_{u_0(y)}k(y-x+\delta)\,dy+\int_0^{x+b-\delta}u(y)k(y-x+\delta)\,dy-\int_{x-a-\delta}^{x+b-\delta}\underbrace{u(x-\delta)}_{=u_0(x-\delta)}k(y-x+\delta)\,dy = f(x-\delta,u(x-\delta))$$

$$\Leftrightarrow \int_0^{x+b-\delta}u(y)k(y-x+\delta)\,dy = \underbrace{-\int_{x-a-\delta}^{0}u_0(y)k(y-x+\delta)\,dy+\int_{x-a-\delta}^{x+b-\delta}u_0(x-\delta)k(y-x+\delta)\,dy+f(x-}_{\text{known function, denote by } \tilde{f}(x-\delta,u(x-\delta))}$$

Hence, we need to solve for u which satisfies:

$$\int_0^{x+b-\delta}u(y)k(y-x+\delta)\,dy = \tilde{f}(x-\delta,u(x-\delta)), \quad \forall x\in[0,\delta]$$

Differentiate w.r.t. x both sides (also assume that $\tilde{f}$ only depends on $x-\delta$, for now, this means that u only depends on x). We obtain:

$$u(x+b-\delta)k(b) = \tilde{f}'(x-\delta) \Rightarrow u(x+b-\delta) = \frac{1}{k(b)}\tilde{f}'(x-\delta) \Rightarrow u(x+b-\delta) = \frac{1}{k(b)}\tilde{f}'(x-\delta), \quad x\in[0,\delta]$$

It seems that one-sided kernel works for $a=0$? Then,

$$\int_0^{x+b-\delta}u(y)k(y-x+\delta)\,dy = \underbrace{-\int_{x-\delta}^{0}u_0(y)k(y-x+\delta)\,dy+\int_{x-\delta}^{x+b-\delta}u_0(x-\delta)k(y-x+\delta)\,dy+f(x-\delta,u(x}_{\text{known function, denote by } \tilde{f}(x-\delta,u(x-\delta))}$$

(3) The most important generalization is to allow f to depend on u.

IF we have (1) with $\tilde{f}(x-\delta,u(x-\delta))$, then: PR: This is like an "explicit continuation". We should make it into a lemma.

(5) $$u(x)k(\delta) = \tilde{f}_x(x-\delta,u(x-\delta)) + \tilde{f}_u(x-\delta,u(x-\delta))u'(x-\delta)$$

Assume $\tilde{f}(x-\delta,u(x-\delta)) = cu(x-\delta)$, $c\in\mathbb{R}$. Then:

$$u(x) = \frac{1}{k(\delta)}cu'(x-\delta).$$

Choosing $u(x) = e^{\lambda x}$ yields a solution:

$$\frac{1}{k(\delta)} c e^{\lambda x - \lambda\delta} = \lambda \Rightarrow \frac{c}{k(\delta)} e^{-\lambda\delta} = \lambda \Rightarrow \text{we can show existence of such a } \lambda.$$

Question on this: If $\tilde{f}_x(x-\delta, u(x-\delta))$ depends on u and u is a function of x, shouldn't $u(x) = \frac{2}{k(\delta)} c u'(x-\delta)$?

(4) With this explicit solution in hand we can study the convergence of solution to nonlocal IVPs to local counterpart (for this we will need the correct scaling of the kernel). The numerical team can work out solutions even for more general kernels and nonlinearities f.

9. Brainstorm

9.1. Potential Research Directions/Questions:

(1) Find kernels where the solution to the nonlocal differential equation converges to the solution to the classical differential equation. PR: Wait a bit on this. You have quite a bit above to work with.

(2) Work backwards to find the optimal kernel for a given DEQ and initial conditions? (i.e. take a subset of points from the solution to the corresponding classical DEQ and try to find a kernel which could get the solution to the nonlocal DEQ system of equations as close as possible to matching those points) PR: So it looks like an initial condition set on an interavl 2δ may give solvability to the system. First, see if that checks out. If it does, then we can try to prove some theorems, implement a more general set up etc. Again, there is quite abit to do above

(3) Think about other approaches to providing initial conditions. PR: Lster :)

9.2. Thoughts and Observations (discussion).

Note: Extending the domain of u that we're solving for (i.e. now also solving for $u_{k+1}, u_{k+2}, ...u_{k+m}$) simply adds extra equations to the system of equations. This does not affect the previously determined solution set since all of the previous equations remain intact and can stand alone as it's own system. Also, this system is not difficult to determine by hand.

If we take u_{-2}, u_{-1}, u_0, u_1 as our initial conditions for horizon $= 2$, we can immediately solve for u_2 as

$$u_2 = \frac{-a_{-2}u_{-2} - a_{-1}u_{-1} - a_0 u_0 - a_1 u_1 + f(u_0, x_0)}{a_2}$$

In general for u_k and horizon δ, (I believe) we have the recurrence relation with initial condition $u_{-2\delta}, \dots u_{2\delta-1}$ seen below:

$$u_k = \frac{-a_{k-1}u_{k-1} - a_{k-2}u_{k-2} \cdots + f(u_{k-\delta}, x_{k-\delta}) - a_{k-\delta}u_{k-\delta} \cdots - a_{k-2\delta}u_{k-2\delta}}{a_k}$$

From our initial conditions we can immediately solve for $u_{2\delta}$ and recursively solve for any u_k. This should work given any $\mathcal{D}_k u = f(u, x)$. This makes programming a solver of a nonlocal DEQ easier.

Say we want our solution set u_i of the nonlocal DEQ to match with the solution given to us by the corresponding classical DEQ, then perhaps we can derive/compute a kernel

$(a_{-\delta}, \dots a_{\delta-1})$ (or maybe for some reason initial conditions idk) which can immediately yield this result for u_i.

Side note: with our system of equations, we could solve the system immediately even if we had a nonlocal DEQ of the form $\mathcal{D}_k u_i = f(u_i, \dots u_{i+\delta-1}, x_k)$ -AB

9.2.1. *State what the interval is* $[x_{-2}, +\infty)$. Based on the above discussion[5.2.], we can change the assumptions presented at the beginning of this article to consider only finite intervals.

If we take u_{-2}, u_{-1}, u_0, u_1 as our initial conditions for $horizon = 2$, we can recursively solve all u_k in $[x_{-2}, +\infty)$. But the error here depends on what the kernel function looks like.

9.3. **Computations. Matlab:** https://drive.mathworks.com/sharing

In this early stage of computations of the above systems of equations, I find that the computed solution to $\mathcal{D}_k u = 0$ with initial conditions $u_{-2}, u_{-1}, u_0, u_1 = 5$ and constant kernel yields a near "correct solution" with several outlying jumps. -AB

(Emrys King, July 19) Working on creating a function in MatLab that takes inputs of the nonlocal derivative, the initial conditions, the kernel, the left and right bounds of an interval, and the number of equations for the system of equations method. The function then sets up the system of equations, solves it, and plots the nonlocal solution alongside the classical solution.

(Emrys King, July 22) Code is currently not working — my setup of the kernel coefficients/system of equations has some dimensional disagreement. Here is a link to matlab code if anyone else has comments/advice on it: https://github.com/egkegk/polymathjr. I will return to it tomorrow and keep working on it!

9.4. **Articles/Papers/Readings.**