Numerical analysis

Coursework 2

Stefan A. Obada

so1118

March 2020

Abstract

This paper solves the questions from the 2^{nd} Coursework. I study a recurrence relation for the p^{th} derivative of Chebyshev polynomials and solve the differential equation $\ddot{y}(x) - xy(x) = 0$ using this result.

Contents

Problem 1

Definition: Chebyshev polynomials are defined by the relation

$$T_n(x) = cos(n * arccos(x)) \tag{1}$$

Note: We use this definition, as in the lectures. However, this is just one of the 2 forms of Chebyshev polynomials. Moreover, this one holds only for $x \in (-1, 1)$.

Recurrence for derivatives

To derive a recurrence relation for the p^{th} derivative we will firstly derive the differential equation that is verified by T_n and we will proceed by using Leibniz Lemma of differentiation.

Lemma 1: T_k verifies one of the Sturm-Liouville differential equations, which is:

$$(1-x^2)\ddot{T}_k(x) = k^2 T_k(x) + x\dot{T}_k(x) \tag{2}$$

Proof: Differentiate $T_k(x)$ to obtain:

$$\dot{T}_k(x) = k \cdot \frac{sin(k \cdot arccos(x))}{\sqrt{1-x^2}} \tag{3}$$

$$\ddot{T}_k(x) = \frac{d}{dx}\dot{T}_k(x) = k \cdot \frac{cos(k \cdot arccos(x))\frac{k\sqrt{1-x^2}}{\sqrt{1-x^2}} - sin(k \cdot arccos(x)) \cdot \frac{-2x}{2\sqrt{1-x^2}}}{(1-x^2)} =$$

$$= \frac{k^2 T_k(x) + x\dot{T}_k(x)}{1-x^2} \Rightarrow \underline{(1-x^2)\ddot{T}_k(x) = k^2 T_k(x) + x\dot{T}_k(x)}$$

Lemma 2: (Leibniz formula) For f and g real functions n times differentiable, we have:

$$(fg)^{(n)} = \sum_{k=0}^{n} \binom{n}{k} \cdot f^{(n-k)} g^{(k)}$$

Lemma 3: The recurrence relation between the derivatives of $T = T_n(x)$ is given by:

$$(1-x^2)T^{(p)} = (2p-3) \cdot xT^{(p-1)} + [k^2 + (p-2)^2] \cdot T^{(p-2)} \tag{4}$$

Proof: Firstly, differentiate (2) from **Lemma 1**, $p-2$ times. Note that all the terms involved are infinite times differentiable.

$$[(1-x^2)\ddot{T}]^{(p-2)} = k^2 T^{(p-2)} + [x\dot{T}]^{(p-2)} \tag{5}$$

We proceed by applying **Lemma 2** on LHS and RHS. Note that $(1-x^2)^{(p)} = 0, \forall p \geq 3$ and $x^{(p)} = 0, \forall p \geq 2$.

$$
\begin{aligned}
[(1-x^2)\ddot{T}]^{(p-2)} &= \binom{p-2}{0}(1-x^2)T^{(p)} + \binom{p-2}{1}(-2x)T^{(p-1)} + \binom{p-2}{2}(-2)T^{(p-2)} \\
&= (1-x^2)T^{(p)} - (p-2)2x \cdot T^{(p-1)} - (p-2)(p-3)T^{(p-2)} \\
[x\dot{T}]^{(p-2)} &= \binom{p-2}{0}xT^{(p-1)} + \binom{p-2}{1}T^{(p-2)} = xT^{(p-1)} + (p-2)T^{(p-2)}
\end{aligned}
$$

By plugging these 2 relations into (5) and rearranging results in:

$$
(1-x^2)T^{(p)} = [(p-2)2x + x]T^{(p-1)} + [k^2 + (p-2)(p-3) + p - 2]T^{(p-2)}
$$

which finishes the proof.
Remark 1: $T_n^{(n+1)}(x) = 0$ and $T_n^{(p)}(x) \neq 0, \forall p \leq n$
Remark 2: The initial terms $T_n^{(0)}$ and $T_n^{(1)}$ are defined as in (1) and (3).

Evaluation at Gauss-Lobatto points

We simply evaluate (4) at $x_j = cos(\frac{j\pi}{N})$:

$$
sin^2(\frac{j\pi}{N})^{\cdot}T^{(p)} = (2p-3) \cdot cos(\frac{j\pi}{N}) \cdot T^{(p-1)} + [k^2 + (p-2)^2] \cdot T^{(p-2)} \tag{6}
$$

Where the initial terms are:

$$
T_k^{(0)}(x_j) = cos(\frac{kj\pi}{N}) \; and \; T_k^{(1)}(x_j) = \frac{k \cdot sin(\frac{kj\pi}{N})}{sin(\frac{j\pi}{N})} \tag{7}
$$

Problem 2

Idea

Step 0: Rescale the initial differential equation on the interval $[-1, 1]$ using the function $u(x) = y(40x)$.

$$\ddot{y}(x) - xy(x) = 0, \quad -40 \leq x \leq 40, \quad y(40) = 0, \quad \dot{y}(-40) = 1$$

is equivalent (due to linearity of the equation and basic differentiation) to:

$$\ddot{u}(x) - 40^3 xu(x) = 0, \quad -1 \leq x \leq 1, \quad u(1) = 0, \quad \dot{u}(-1) = 40 \tag{8}$$

Step 1: Write the Chebyshev expansion for $u(x)$ (up to $N < \infty$) and differentiate to obtain a relation for $\ddot{u}(x)$:

$$\begin{aligned} u(x) &= \sum_{k=0}^{N} a_k T_k(x) \\ \ddot{u}(x) &= \sum_{k=0}^{N} a_k \ddot{T}_k(x) \end{aligned}$$

where T_k is the k^{th} Chebyshev polynomial and $\dot{T}_k$ its derivative.

Step 2: Evaluate these relations at a of Gauss-Lobatto points: $x_j = cos(\frac{j\pi}{N}), \forall j = 0, 1, ..., N$ using (1), (2) and (3):

$$\begin{aligned} u(x_j) &= \sum_{k=0}^{N} a_k \cdot T_k(x_j) \\ \ddot{u}(x_j) &= \sum_{k=0}^{N} a_k \cdot [\frac{k^2}{1 - x_j^2} T_k(x_j) + \frac{x_j}{1 - x_j^2} \dot{T}_k(x_j)] \end{aligned}$$

Step 3: Rewrite the differential equation $\ddot{u}(x) - 40^3 xu(x) = 0$ into matrix form:

$$\ddot{u}(x_j) - 40^3 x_j u(x_j) = \sum_{k=0}^{N} a_k \cdot [(\frac{k^2}{1 - x_j^2} - 40^3 x_j) \cdot T_k(x_j) + \frac{x_j}{1 - x_j^2} \dot{T}_k(x_j)] = 0$$

Therefore let $D_{jk} = [(\frac{k^2}{1-x_j^2} - 40^3 x_j) \cdot T_k(x_j) + \frac{x_j}{1-x_j^2} \dot{T}_k(x_j)]$, for all $j, k \in (0, 1, ..., N)$.

Step 4: Introduce boundary conditions $u(1) = 0$ and $\dot{u}(-1) = 40$. Note that these are equivalent to: $u(x_0) = 0$ and $\dot{u}(x_N) = 40$, which is therefore equivalent to changing the first and the last rows of D accordingly:

$$D_{0,k} = [T_k(x_0)]$$
$$D_{N,k} = [\dot{T}_k(x_N)]$$

Finally, the equation reduces to solving $\mathbf{a} = (a_1, a_2..., a_N)^T$ in:

$D\mathbf{a} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 40 \end{bmatrix}$ which can be performed via several computational methods.

Remark 1: $y(x)$ can be approximated via: $y(x) = u(\frac{x}{40}) = \sum_{k=0}^{N} a_k T_k(\frac{x}{40})$, where T_k are the Chebyshev polynomials.

Remark 2: We need to work with the polynomial form of T_k computed recursively (because we need to compute $\dot{T}_k(-1)$).

Implementation (Python)

```
import numpy as np
import matplotlib.pyplot as plt

# Set N
N = 20

# Create vector of evaluations [0, ..., 0, 40]
ev = np.zeros(shape=(N+1,1))
ev[-1] = 40

# Define functions T and its derivative, T_dot recursive
def T(x, k):
    if k==0:
        # 1st Chebyshev polynomial
        return 1
    elif k==1:
        # 2nd Chebyshev polynomial
        return x
    else:
        # Recurrence formula
        return 2*T(x, k-1)*x - T(x, k-2)

def T_dot(x, k):
    if k==0:
        # 1st Chebyshev poly derivative
```

```
        return 0
    elif k==1:
        # 2nd Chebyshev poly derivative
        return 1
    else:
        # Recurrence formula for derivatives
        return 2*T(x, k-1)+2*x*T_dot(x, k-1)-T_dot(x, k-2)

# Create the Gauss-Lobatto linear space, i.e. x_j
X = []
for j in range(N+1):
    X.append(np.cos(j * np.pi / N))

# Create D - matrix function, according to Step 3
def D_jk(j, k):
    return ( (k**2 / (1 - X[j]**2)) - 40**3 * X[j] ) * T(X[
                                j], k) + ( X[j] / (1 - X[j]**2
                                ) ) * T_dot(X[j], k)

D = np.zeros(shape=(N+1, N+1)) # Matrix of zeros

for j in range(N+1):
    if j==0:
        # The 1st row (for boundary condition)
        for k in range(N+1):
            D[j, k] = T(X[j], k)
    elif j==N:
        # The last row (for boundary condition)
        for k in range(N+1):
            D[j, k] = T_dot(X[j], k)
    else:
        # Matrix body
        for k in range(N+1):
            D[j, k] = D_jk(j, k)

# Linear solver
D = np.array(D)
a = np.linalg.inv(D).dot(ev)

# Define the solution function y as in Remark 1
def y(x):
    Tk_vector = np.array([T(x/40, k) for k in range(N+1)])
    return Tk_vector.dot(a)[0]

# Plot the solution
linsp = np.linspace(start=-40, stop=40, num=300)
plt.plot(linsp, [y(x) for x in linsp], c='red')
```

Solution plot

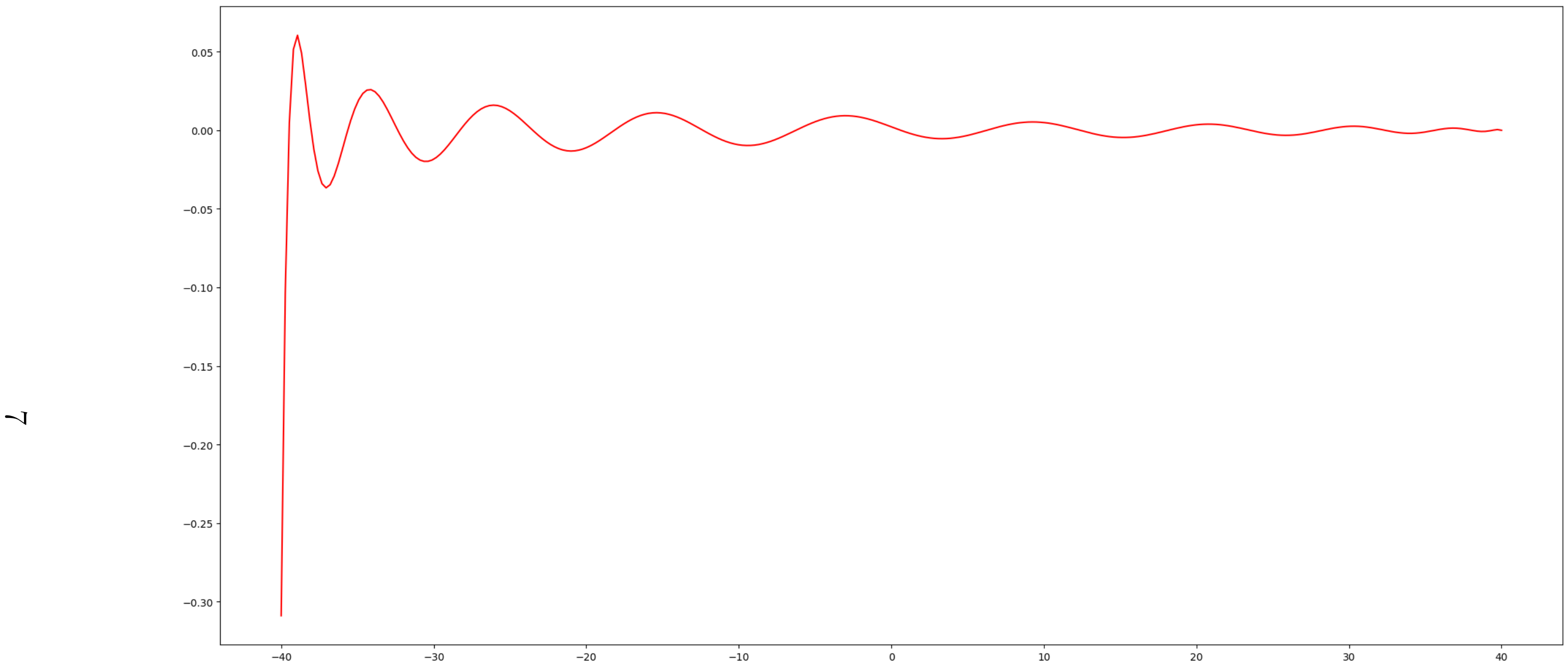

Figure 1: x vs. $y(x)$