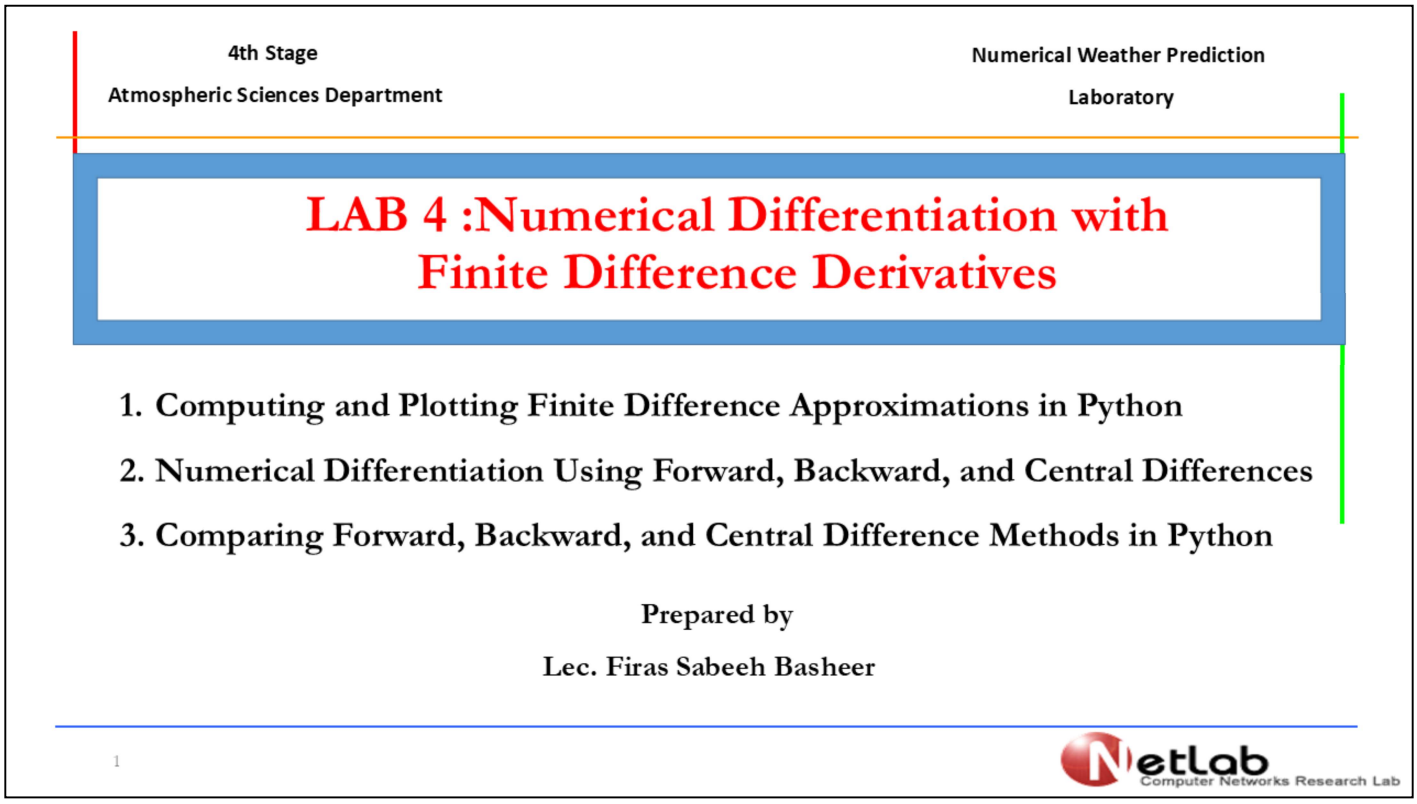


LAB 4 :Numerical Differentiation with Finite Difference Derivatives

التفاضل العددي باستخدام طريقة الفروقات المحددة

1. حساب ورسم تقريبات الفروقات المحددة باستخدام بايثون
2. التفاضل العددي باستخدام الفروقات الأمامية والعكسية والمركزية
3. مقارنة طرائق الفروقات الأمامية والعكسية والمركزية في بايثون

❑ **The Finite Difference Method** is a numerical technique used to approximate the derivatives of a function by evaluating its values at discrete, closely spaced points.

Instead of using analytical calculus (like taking limits or derivative rules) to find an exact slope, the method replaces the continuous derivative operator with **algebraic difference quotients**.

❑ **Why Use It?**

1- Experimental Data: In real-world applications, you often only have a table of measured data points (e.g., time vs. velocity from a sensor) rather than a smooth mathematical formula you can differentiate by hand.

2- Complex Functions: Some functions are analytically impossible or computationally too difficult to differentiate directly.

NetLab Computer Networks Research Lab

❑**طريقة الفروقات المحددة (Finite Difference Method)** هي تقنية عددية تستخدم لتقريب مشتقات دالة رياضية من خلال تقييم قيمها عند نقاط منفصلة ومتقاربة بمسافات صغيرة.

❑ وبدلاً من استخدام حساب التفاضل والتكامل التحليلي (مثل قواعد الاشتقاق المباشر) لإيجاد الميل بدقة، تقوم هذه الطريقة استبدال مشتقة الدالة بـ **معادلات جبرية بسيطة تعتمد على التغير في القيم**.

❑**لماذا نستخدم هذه الطريقة؟**

1- البيانات التجريبية: في التطبيقات الواقعية، غالباً ما نملك جدولاً يحتوي على قراءات و مقاييس مجدولة (مثل السرعة مقابل الزمن من مستشعر) وليس معادلة رياضية سلسة يمكن اشتقاقها يدوياً.

2- الدوال المعقدة: بعض الدوال يصعب أو يستحيل اشتقاقها بالطرق الجبرية التقليدية.

❑ Theoretical Foundation

Finite difference methods are derived from the **Taylor series expansion**. The lab typically introduces three primary formulas to approximate the first derivative at a point x using a step size h:

1. Forward Difference Formula

Uses the current point and a forward point:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

2. Backward Difference Formula

Uses the current point and a previous point:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

NetLab Computer Networks Research Lab

المفاهيم الأساسية

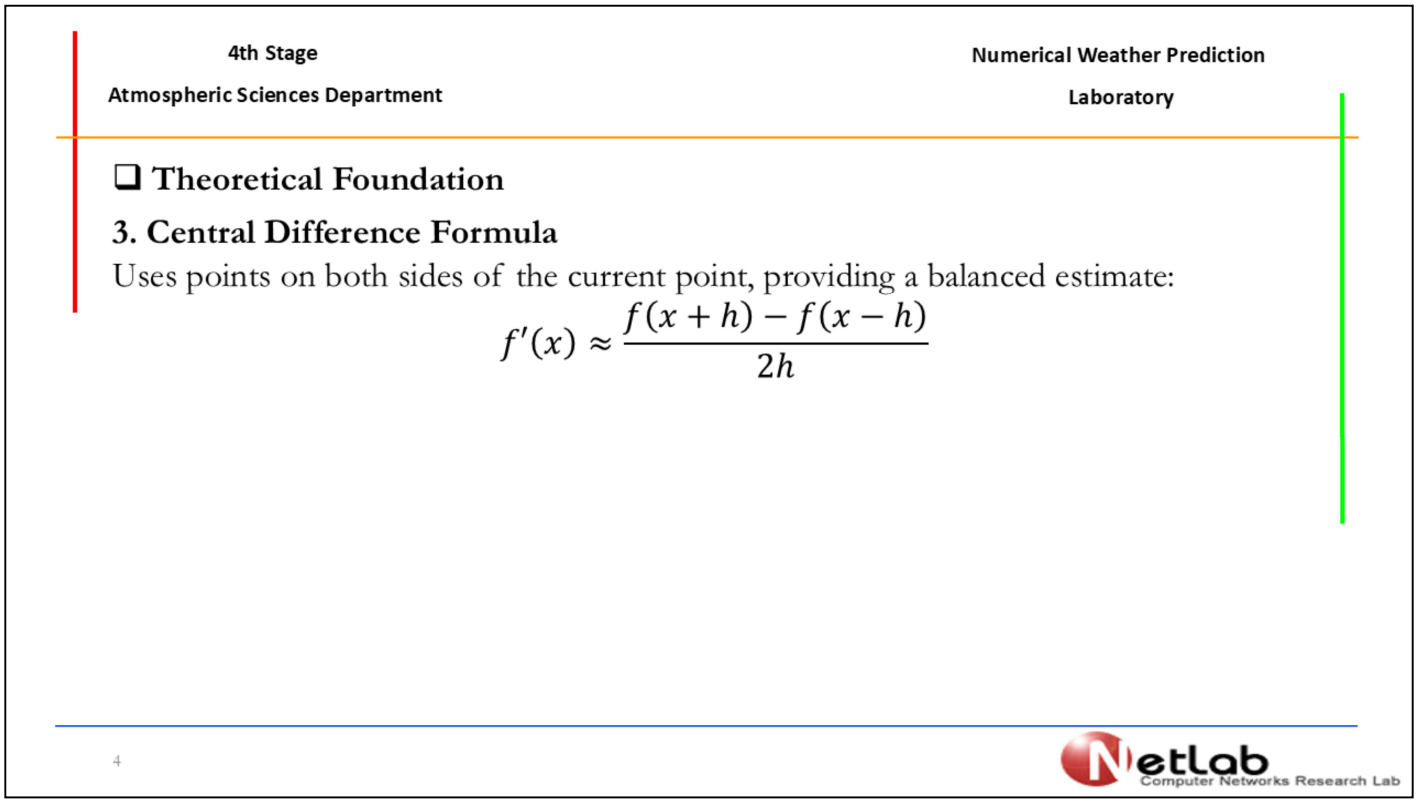
❑ تُشتق طريقة الفروقات المحددة من متسلسلة تايلور (Taylor series expansion). يقدم البرنامج عادةً ثلاث صيغ رئيسية لتقريب المشتقة الأولى عند نقطة x باستخدام حجم خطوة h:

1. **صيغة الفروقات الأمامية (Forward Difference Formula)** تستخدم النقطة الحالية ونقطة أمامية:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

2. **صيغة الفروقات العكسية (Backward Difference Formula)** تستخدم النقطة الحالية ونقطة سابقة:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$



3. صيغة الفروقات المركزية (Central Difference Formula) تستخدم النقاط على كلا الجانبين للنقطة الحالية، مما يوفر تقديرًا متوازنًا:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

LAB 4 : Finite Difference Derivatives

1. Computing and Plotting Finite Difference Approximations in Python

- We will now implement these finite difference methods in Python. To evaluate their accuracy, we will apply them to a function whose exact derivative is known. Specifically, we will use the function $f(t) = \sin(t)$,

```
[1]: import numpy as np
     from matplotlib import pyplot as plt
```

```
[2]: dt = 0.2

     t = np.linspace(-2, 4, 60)
     f = np.sin(t)
```

5

NetLab Computer Networks Research Lab

1. حساب ورسم تقريبات الفروقات المحددة باستخدام بايثون

- سنقوم الآن بتطبيق طريقة الفروقات المحددة باستخدام لغة بايثون .ولتقييم دقتها، سنستخدم دالة نعرف مشتقتها الحقيقية مسبقًا .على وجه التحديد، سنستخدم الدالة $f(t)=\sin(t)$

LAB 4 : Finite Difference Derivatives

2. The calculations will be performed over the interval $t = -2$to $t = 4$using a finely spaced grid,

[1]:
```
import numpy as np
from matplotlib import pyplot as plt
```

[2]:
```
dt = 0.2

t = np.linspace(-2, 4, 60)
f = np.sin(t)
```


NetLab Computer Networks Research Lab

2. سيتم إجراء الحسابات على المجال $t=-2$ إلى $t=4$ باستخدام شبكة زمنية ذات دقة عالية،

LAB 4 : Finite Difference Derivatives

3. whose analytical derivative is

[3]:
```
# Exact Derivative
dfdt = np.cos(t)
```


NetLab Computer Networks Research Lab

- التي تكون مشتقتها التحليلية $df/dt=\cos$ (t).

LAB 4 : Finite Difference Derivatives

4. Now, we will plot the function and its exact derivative for comparison.

[4]:
```
# plotting commands

plt.plot(t, f, 'k--', linewidth=1.2)
plt.grid(True)

plt.plot(t, dfdt, 'k', linewidth=3)

plt.legend(['Function', 'Exact Derivative'], fontsize=14)

plt.xlim(-2, 4)
plt.ylim(-1.5, 1.5)

plt.show()
```

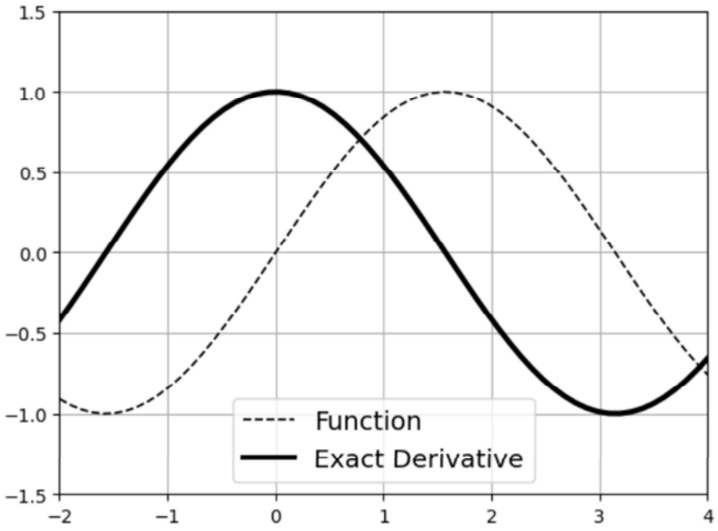

NetLab Computer Networks Research Lab

4. سنقوم الآن برسم الدالة ومشتقتها الحقيقية للمقارنة.

LAB 4 : Finite Difference Derivatives

4. Next, we will plot the function and its exact derivative for comparison.

The following plot displays the function $f(t) = \sin(t)$ and its exact derivative $\frac{df}{dt} = \cos(t)$over time. This comparison serves as a reference for evaluating the forward, backward, and central difference approximations.



9

NetLab Computer Networks Research Lab

سنقوم الآن برسم الدالة ومشتقتها الحقيقية للمقارنة.

يوضح الشكل التالي الدالة $f(t)=\sin(t)$ ومشتقتها الحقيقية $df/dt=\cos(t)$ بدلالة الزمن .ويُستخدم هذا التمثيل البياني كمرجع لمقارنة وتقييم دقة تقريبات الفروقات الأمامية والعكسية والمركزية في حساب المشتقات العددية.

LAB 4 : Finite Difference Derivatives

5. Now, we compute the numerical derivative approximations using a time step of $\Delta t = 0.2$.
The forward difference approximation (F),
backward difference approximation (B),
and central difference approximation (C) are calculated for the entire time vector t.

[5]:
```
# Forward Difference Approximation
dfdtF = (np.sin(t + dt) - np.sin(t)) / dt

# Backward Difference Approximation
dfdtB = (np.sin(t) - np.sin(t - dt)) / dt

# Central Difference Approximation
dfdtC = (np.sin(t + dt) - np.sin(t - dt)) / (2 * dt)
```

10

NetLab Computer Networks Research Lab

5. نحسب الان تقريبات المشتقة العددية باستخدام خطوة زمنية مقدارها Δt=0.2.
يتم حساب تقريبات الفروقات الأمامية (F)
والعكسية (B)
والمركزية (C) لجميع نقاط متجه الزمن t.

LAB 4 : Finite Difference Derivatives

[5]:
```
# Forward Difference Approximation
dfdtF = (np.sin(t + dt) - np.sin(t)) / dt

# Backward Difference Approximation
dfdtB = (np.sin(t) - np.sin(t - dt)) / dt

# Central Difference Approximation
dfdtC = (np.sin(t + dt) - np.sin(t - dt)) / (2 * dt)
```

The forward difference uses $f(t + \Delta t)$and $f(t)$,
the backward difference uses $f(t)$and $f(t - \Delta t)$,
while the central difference uses both $f(t + \Delta t)$and $f(t - \Delta t)$.

11

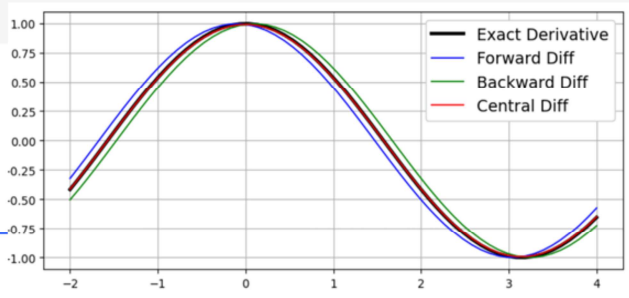
NetLab Computer Networks Research Lab

تستخدم طريقة الفروقات الأمامية القيمتين $f(t+\Delta t)$و$f(t)$،
بينما تستخدم طريقة الفروقات العكسية القيمتين $f(t)$و$f(t-\Delta t)$،
أما طريقة الفروقات المركزية فتستخدم $f(t+\Delta t)$ و$f(t-\Delta t)$.

LAB 4 : Finite Difference Derivatives

6. These approximations are then compared with the exact derivative to evaluate their accuracy.

```
[6]: plt.figure(figsize=(20, 4))
     plt.subplot(1, 2, 1)
     plt.plot(t, dfdt,  'k', linewidth=3)
     plt.plot(t, dfdtF, 'b', linewidth=1.2)   # Forward Difference
     plt.plot(t, dfdtB, 'g', linewidth=1.2)   # Backward Difference
     plt.plot(t, dfdtC, 'r', linewidth=1.2)   # Central Difference
     plt.legend(['Exact Derivative','Forward Diff','Backward Diff','Central Diff'],fontsize=14,framealpha=1.0)
     plt.grid(True)
     plt.show()
```



12

NetLab Computer Networks Research Lab

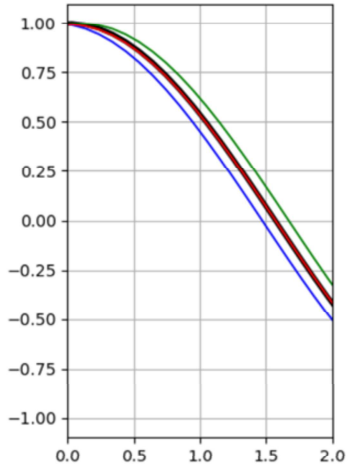
6. نعمل الان مقارنة هذه التقريبات بالمشتقة الحقيقية لتقييم دقة كل طريقة.

❑ **يوضح الشكل المشتقة الحقيقية** $\cos(t)$ **مع تقريبات الفروقات الأمامية والعكسية والمركزية. تُظهر النتائج أن المنحنيات العددية الثلاثة تتوافق بشكل كبير مع المشتقة الحقيقية، مما يدل على أن هذه الطرق تعطي تقديرات جيدة للمشتقة عند استخدام خطوة زمنية** $\Delta t=0.2$.

LAB 4 : Finite Difference Derivatives

7. When zooming in on the plot, the forward difference tends to underestimate the derivative, while the backward difference tends to overestimate it.

```
[7]: # Zoomed-in plot
     plt.subplot(1, 2, 2)
     plt.plot(t, dfdt,  'k', linewidth=3)
     plt.plot(t, dfdtF, 'b', linewidth=1.2)
     plt.plot(t, dfdtB, 'g', linewidth=1.2)
     plt.plot(t, dfdtC, 'r', linewidth=1.2)
     plt.grid(True)
     plt.xlim(0, 2)
     plt.show()
```



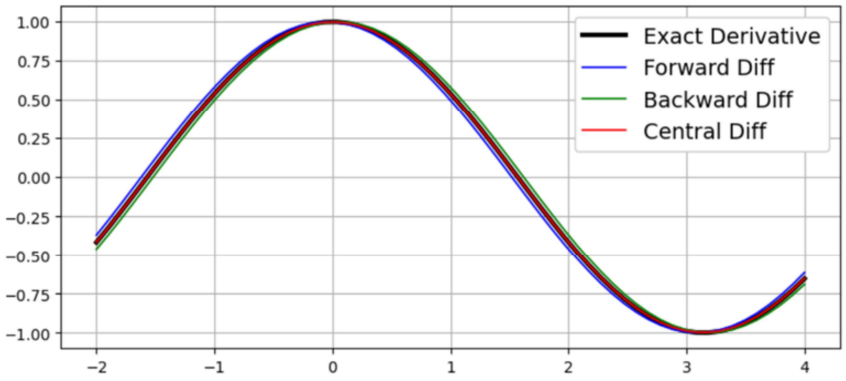
13

NetLab Computer Networks Research Lab

7. وعند تكبير الشكل، نلاحظ أن طريقة الفروق الأمامية تميل إلى التقليل قليلًا من قيمة المشتقة، بينما تميل طريقة الفروقات العكسية إلى المبالغة قليلًا في تقديرها. أما طريقة الفروقات المركزية فتتطابق تقريبًا مع المشتقة الحقيقية، مما يبرز دقتها العالية. وتعود الفروقات البسيطة المتبقية أساسًا إلى انحناء الدالة وحجم الخطوة الزمنية المحدد Δt.

LAB 4 : Finite Difference Derivatives

8. Reducing the time step from $\Delta t = 0.2$to $\Delta t = 0.1$further improves the accuracy of the finite-difference approximations, with the central difference converging to the exact derivative more rapidly than the forward and backward differences.



14

NetLab Computer Networks Research Lab

8. يؤدي تقليل خطوة الزمن من Δt=0.2 إلى Δt=0.1 إلى تحسين دقة تقريبات الفروقات المحددة، حيث تتقارب النتائج بشكل أكبر نحو المشتقة الحقيقية، خاصةً في طريقة الفروقات المركزية.

LAB 4 : Finite Difference Derivatives

- Example: Applying Forward, Backward, and Central Difference Methods to NetCDF Data

[4]:

```
import matplotlib.pyplot as plt
import numpy as np
import xarray as xr
ds = xr.open_dataset('data_sfc.nc')
ds.data_vars
```

[4]:

```
Data variables:
    u10      (valid_time, latitude, longitude) float32 ...
    v10      (valid_time, latitude, longitude) float32 ...
```

15

NetLab Computer Networks Research Lab

مثال :تطبيق طريقة الاشتقاق العددي باستخدام الفروقات المحددة الأمامية والعكسية والمركزية على بيانات NetCDF (ملفات nc-file)

في هذا المثال قراءة بيانات من ملف NetCDF، واختيار سلسلة زمنية لمتغير سرعة الرياح u10 عند نقطة محددة، ثم تطبيق طريقة الفروقات المحددة الأمامية والعكسية والمركزية لحساب المشتقة الزمنية للمتغير.

اولا : قراءة أسماء المتغيرات

LAB 4 : Finite Difference Derivatives

- Example: Applying Forward, Backward, and Central Difference Methods to NetCDF Data

```
[ ]: import matplotlib.pyplot as plt
     import numpy as np
     import xarray as xr
     ds = xr.open_dataset('data_sfc.nc')
     # Select nearest grid point
     f = ds.u10.sel(longitude=44.4,latitude=33.3,method='nearest').values[0:240]
     # Time values
     time = ds.valid_time.values[0:240]
     # Convert time to hours since first record
     t = (time - time[0]) / np.timedelta64(1, 'h')
```

16

NetLab Computer Networks Research Lab

مثال :تطبيق طريقة الاشتقاق العددي باستخدام الفروقات المحددة الأمامية والعكسية والمركزية على بيانات NetCDF (ملفات nc-file)

قراءة ملف **NetCDF**باستخدام مكتبة **xarray**، ثم اختيار أقرب نقطة شبكية إلى الموقع المحدد بخط طول ٥^[44.4]وخط عرض ٥^[33.3]واستخراج قيم سرعة الرياح عند ارتفاع 10أمتار (**u10**)لأول 240نقطة زمنية .بعد ذلك يتم استخراج قيم الزمن وتحويلها من صيغة التاريخ والوقت إلى عدد الساعات من بداية الفترة الزمنية، لتطبيق طريقة الفروقات المحددة وحساب المشتقة الزمنية للمتغير.

LAB 4 : Finite Difference Derivatives

[7]:

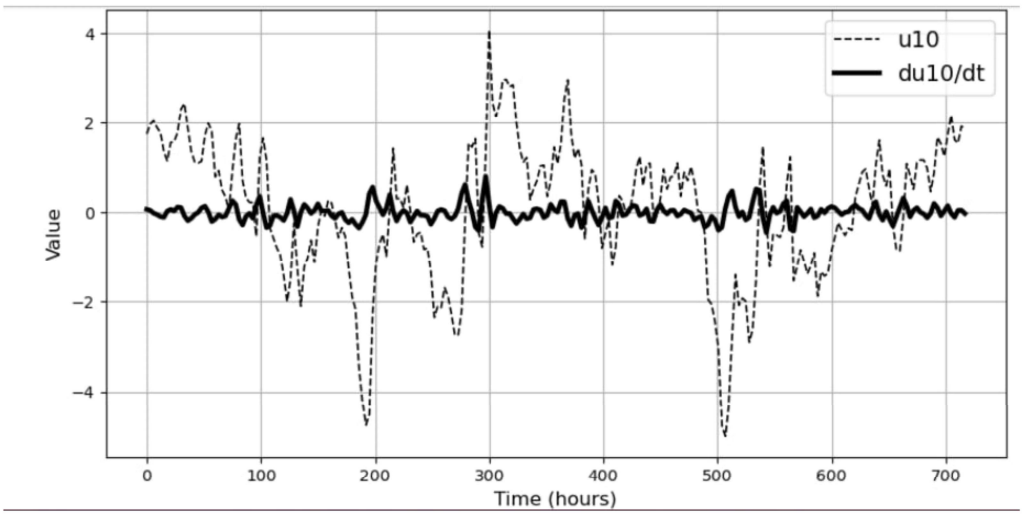
```
# Exact derivative not available for observational/model data
# Use numerical derivative instead
dfdt = np.gradient(f, t)
# Plotting
plt.figure(figsize=(10, 5))
plt.plot(t, f, 'k--', linewidth=1.2)
plt.grid(True)
plt.plot(t, dfdt, 'k', linewidth=3)
plt.legend(['u10', 'du10/dt'], fontsize=14)
plt.xlabel('Time (hours)', fontsize=12)
plt.ylabel('Value', fontsize=12)
plt.show()
```

17

NetLab Computer Networks Research Lab

- يتم حساب **المشتقة العددية** لمتغير سرعة الرياح u10 بالنسبة للزمن باستخدام الدالة np.gradient(). بعد ذلك يتم رسم السلسلة الزمنية الأصلية للرياح والمشتقة الزمنية لها على الشكل نفسه للمقارنة، مما يوضح كيفية تغير سرعة الرياح مع الزمن ومعدل هذا التغير عند كل خطوة زمنية.

LAB 4 : Finite Difference Derivatives



18

NetLab Computer Networks Research Lab

- يتم حساب **المشتقة العددية** لمتغير سرعة الرياح u10بالنسبة للزمن باستخدام الدالة np.gradient(). بعد ذلك يتم رسم السلسلة الزمنية الأصلية للرياح والمشتقة الزمنية لها على الشكل نفسه للمقارنة، مما يوضح كيفية تغير سرعة الرياح مع الزمن ومعدل هذا التغير عند كل خطوة زمنية.

LAB 4 : Finite Difference Derivatives

[8]:
```
# Time step (hours)
dt = t[1] - t[0]
# Forward Difference
dfdtF = np.empty_like(f)
dfdtF[:-1] = (f[1:] - f[:-1]) / dt
dfdtF[-1] = np.nan
# Backward Difference
dfdtB = np.empty_like(f)
dfdtB[0] = np.nan
dfdtB[1:] = (f[1:] - f[:-1]) / dt
# Central Difference
dfdtC = np.empty_like(f)
dfdtC[0] = np.nan
dfdtC[-1] = np.nan
dfdtC[1:-1] = (f[2:] - f[:-2]) / (2 * dt)
```

19

NetLab Computer Networks Research Lab

- يتم حساب الفاصل الزمني Δt بين السجلات الزمنية المتتالية، ثم استخدامه لحساب المشتقة الزمنية لمتغير سرعة الرياح u10 بثلاث طرق عددية مختلفة هي:

الفروقات الأمامية (Forward Difference) باستخدام القيم الحالية واللاحقة.

الفروقات العكسية (Backward Difference) باستخدام القيم الحالية والسابقة.

الفروقات المركزية (Central Difference) باستخدام القيم السابقة واللاحقة، وتتميز عادةً بدقة أعلى.

تُستخدم هذه الطرق لتقدير معدل تغير سرعة الرياح مع الزمن ومقارنة أداء كل طريقة في حساب المشتقة العددية.

ملاحظة : np.empty_like(f) وتعني هذه الخطوة إنشاء مصفوفة فارغة بنفس أبعاد و بنية مصفوفة البيانات الأصلية f لتخزين مشتقات الفروقات (الامامية والعكسية والمركزية)

dfdtF[:-1] = (f[1:] - f[:-1]) / dt: تطبيق صيغة الفروقات الأمامية لكل العناصر ما عدا العنصر الأخير، حيث يتم طرح القيمة الحالية من القيمة التي تليها وقسمتها على حجم الخطوة dt.

dfdtF[-1] = np.nan: تعيين القيمة الأخيرة كـ NaN (غير معرفة لعدم وجود نقطة أمامية بعدها تتيح حساب المشتقة الأمامية .

dfdtB[0] = np.nan: تعيين القيمة الأولى كـ NaN لعدم وجود نقطة سابقة قبلها تتيح حساب المشتقة العكسية.

dfdtB[1:] = (f[1:] - f[:-1]) / dt: تطبيق صيغة الفروقات العكسية لجميع العناصر بدءاً من العنصر الثاني وحتى النهاية عن طريق طرح القيمة السابقة من القيمة الحالية وقسمتها على dt.

dfdtC[0] = np.nan و dfdtC[-1] = np.nan: تعيين كل من العنصر الأول والأخير كـ NaN نظراً لأن صيغة الفروق المركزية تتطلب وجود نقطة سابقة ونقطة لاحقة في آن واحد، وهو ما ينعدم عند الأطراف.

dfdtC[1:-1] = (f[2:] - f[:-2]) / (2 * dt): تطبيق صيغة الفروق المركزية على كافة العناصر الداخلية)من العنصر الثاني قبل الأخير (عن طريق حساب الفرق بين القيمة اللاحقة والقيمة السابقة وقسمتها على ضعف حجم الخطوة $(2 \times dt)$ لضمان الحصول على دقة أعلى)الرتبة الثانية(.

LAB 4 : Finite Difference Derivatives

•[11]:

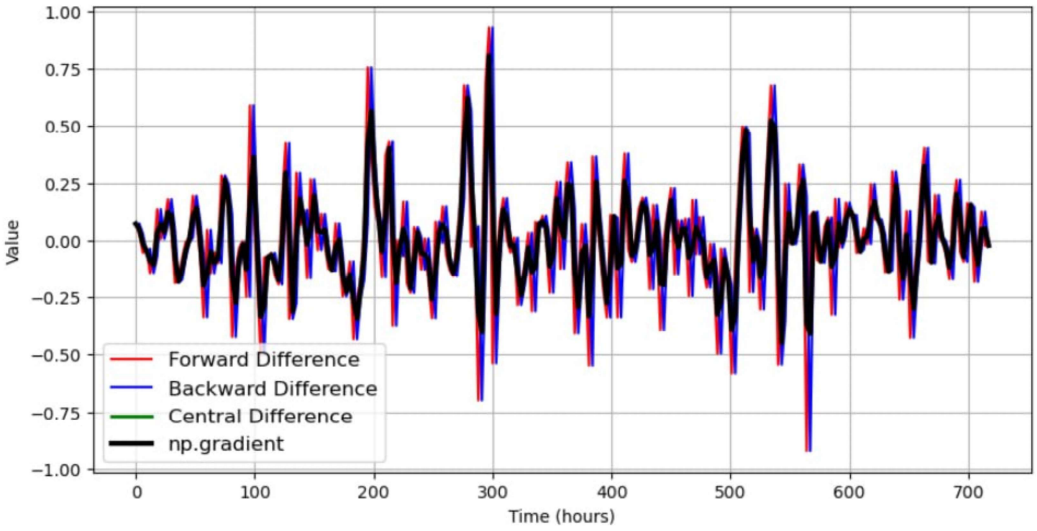
```
# Plot
plt.figure(figsize=(10,5))
plt.plot(t, dfdtF, 'r', linewidth=1.5)
plt.plot(t, dfdtB, 'b', linewidth=1.5)
plt.plot(t, dfdtC, 'g', linewidth=2)
plt.plot(t, dfdt, 'k', linewidth=3)
plt.grid(True)
plt.legend(['Forward Difference','Backward Difference','Central Difference',
            'np.gradient'],fontsize=12)
plt.xlabel('Time (hours)')
plt.ylabel('Value')
plt.show()
```

20

NetLab Computer Networks Research Lab

يتم رسم نتائج طرق الاشتقاق العددي المختلفة، وهي الفروقات الأمامية والعكسية والمركزية ، بالإضافة إلى المشتقة المحسوبة باستخدام الدالة np.gradient(). يتيح هذا الرسم مقارنة نتائج الطرق المختلفة بصريًا وبيان مدى تقاربها أو اختلافها في تمثيل معدل تغير سرعة الرياح مع الزمن، حيث تُظهر الفروقات المركزية عادةً دقة أعلى مقارنةً بالفروقات الأمامية والعكسية.

LAB 4 : Finite Difference Derivatives



21

NetLab Computer Networks Research Lab

يتم رسم نتائج طرق الاشتقاق العددي المختلفة، وهي الفروقات الأمامية والعكسية والمركزية ، بالإضافة إلى المشتقة المحسوبة باستخدام الدالة np.gradient(). يتيح هذا الرسم مقارنة نتائج الطرق المختلفة بصريًا وبيان مدى تقاربها أو اختلافها في تمثيل معدل تغير سرعة الرياح مع الزمن، حيث تُظهر الفروقات المركزية عادةً دقة أعلى مقارنةً بالفروقات الأمامية والعكسية.