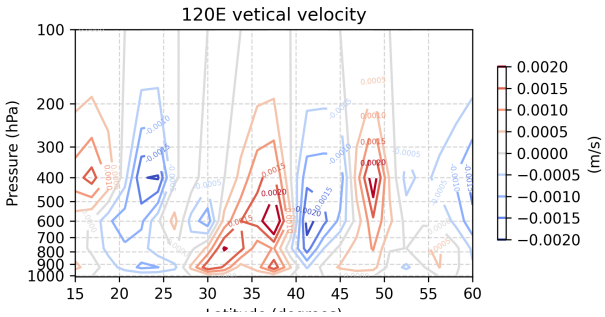
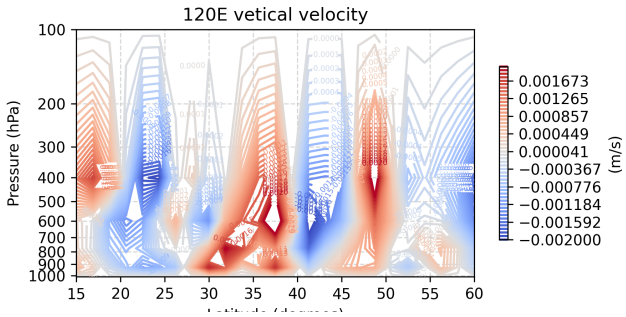
1121 天氣學與天氣分析（下） --- 作業二

- 姓名：林群賀
- 系級：大氣四
- 學號：109601003

執行程式碼

```
$ python3 main.py
```

垂直速度剖面圖

只有分成 10 等分	看出所有垂直速度
	

問題討論：

1. 為何垂直速度要用計算的？計算出來後用途為何？

從綜觀尺度來看，標準的大氣垂直運動速度約為每秒幾公分。然而，通常使用的探空儀器在測量水平風速時，其精確度僅達到每秒一公尺。因此，垂直速度通常是透過計算其他可以實際測量的變數來獲得的。

垂直運動可以影響天氣的晴朗程度，同時也可以通過觀察鋒面、颱風等現象的垂直剖面構造來獲得有關氣象變化的重要信息

2. 此計算方法有何優缺點？

計算所需的變數相對容易取得。

從綜觀尺度來看，水平風大致保持地轉平衡，幾乎沒有輻合或輻散現象，換句話說，∂u/∂x（水平風的x分量梯度）和 ∂v/∂y（水平風的y分量梯度）兩者大致相等但符號相反。因此，水平輻散場主要受到數值較小的非地轉風的貢獻。當在有限差分法中的測量值各自存在約 10% 的誤差時，輻散場的誤差可能很容易達到 100%。

3. 其他計算垂直速度的方法及其優缺點？

使用絕熱法（Adiabatic Method），我們假設在大氣的熱平衡中，非絕熱的影響微不足道，可以忽略不計。

這種方法對水平速度的誤差不太敏感。然而，當系統受到強烈的非絕熱作用，例如潛熱釋放或輻射，絕熱法可能會產生較大的誤差。

程式碼

讀取資料

```
def load_data(
        file_name,
        var,
        nlev,
        nlat,
        mlon,
    ) -> np.ndarray[Any]:
    data = np.fromfile(
        file_name,
        dtype='<f4',
    )
    data = data.reshape(
        var,
        nlev,
        nlat,
        mlon,
    )

    return data

def configure_parameters(
        mlon,
        nlat,
        data
    ) -> tuple[
        np.linspace,
        np.linspace,
        Any,
        Any,
        Any,
        Any,
    ]:
    lon = np.linspace(90, 180, mlon)
    lat = np.linspace(15, 60, nlat)
    h = data[0, :, :, :]
    u = data[1, :, :, :]
    v = data[2, :, :, :]
    t = data[3, :, :, :]

    return (
        lon,
        lat,
        h,
        u,
        v,
```

```
        t,
    )
```

計算散度

```
def count_divergence(
        u,
        v,
        dy,
        nlev,
        nlat,
        mlon,
        lat,
    ) -> np.ndarray[np.float64]:
    divergence = np.zeros(
        [nlev, nlat, mlon]
    )

    for i in range(nlev):
        for j in range(nlat):
            for k in range(mlon):
                dx = dy * np.cos(lat[j] * np.pi / 180)
                if 1 <= j < nlat - 1 and 1 <= k < mlon - 1:
                    # 計算x方向上的差分
                    x_value = (
                        (u[i, j, k + 1] - u[i, j, k - 1]) /
                        (2 * dx)
                    )
                    # 計算y方向上的差分
                    y_value = (
                        (v[i, j + 1, k] - v[i, j - 1, k]) /
                        (2 * dy)
                    )

                    divergence[i, j, k] = x_value + y_value
                else:
                    # 單邊插植
                    # 計算 x 方向上的差分
                    if k == 0:
                        x_value = (
                            (u[i, j, k + 1] - u[i, j, k]) /
                            dx
                        )
                    elif k == mlon - 1:
                        x_value = (
                            (u[i, j, k] - u[i, j, k - 1]) /
                            dx
                        )
                    else:
                        x_value = (
                            (u[i, j, k + 1] - u[i, j, k - 1]) /
```

```
                            (2 * dx)
                        )

                    # 計算 y 方向上的差分
                    if j == 0:
                        y_value = (v[i, j + 1, k] - v[i, j, k]) / dy
                    elif j == nlat - 1:
                        y_value = (v[i, j, k] - v[i, j - 1, k]) / dy
                    else:
                        y_value = (v[i, j + 1, k] - v[i, j - 1, k]) / (2 *
dy)

                    divergence[i, j, k] = x_value + y_value

    return divergence
```

計算垂直速度

```
def count_vertical_speed(
        divergence,
        nlev,
        nlat,
        mlon,
        pressure_values,
    ) -> np.ndarray[np.float64]:

    init_vertical_speed = np.zeros(
        [nlev, nlat, mlon]
    )
    for i in range(nlev):
        for j in range(nlat):
            for k in range(mlon):
                if i == 0:
                    init_vertical_speed[i, j, k] = (
                        divergence[i, j, k] * pressure_values[i]
                    )
                elif i > 0:
                    init_vertical_speed[i, j, k] = (
                        init_vertical_speed[i - 1, j, k] +
                        divergence[i, j, k] * pressure_values[i]
                    )

    # 5 * 25 * 49
    # nlev=5,
    # nlat=25,
    # mlon=49,

    expanded_error = np.zeros(
        [5, 25, 49]
    )
    for i in range(nlev):
```

```
            for j in range(nlat):
                for k in range(mlon):
                    expanded_error[i, j, k] = (
                        init_vertical_speed[4, j, k] / 910
                    )

    # 修正相對渦度
    fixed_divergence = divergence - expanded_error

    # 計算新的垂直風速
    new_vertical_speed =  np.zeros(
        [nlev, nlat, mlon],
        dtype=float,
    )
    for i in range(nlev):
        for j in range(nlat):
            for k in range(mlon):
                if i == 0:
                    new_vertical_speed[i, j, k] = (
                        fixed_divergence[i, j, k] * pressure_values[i]
                    )
                elif i > 0:
                    new_vertical_speed[i, j, k] = (
                        new_vertical_speed[i - 1, j, k] +
                        fixed_divergence[i, j, k] *
                        pressure_values[i]
                    )

    return new_vertical_speed
```

視覺化

```
def check_output_dir() -> None:
    os.makedirs("imgs", exist_ok=True)

def visualize_results(
        factor,
        lat,
    ) -> None:
    levels = [
        1010,
        925,
        775,
        600,
        400,
        100
    ]
    plt.figure(
        figsize=(6, 3),
        dpi=300,
    )
```

```
    var = np.zeros(
        (6, 25)
    )
    var[1:,:] = factor[:, :, 16]

    contour = plt.contour(
        lat,
        levels,
        var,
        cmap='coolwarm',
        levels = np.linspace(-0.002, 0.002, 9)
    )
    plt.title("120E vetical velocity")
    plt.xlabel("Latitude (degrees)")
    plt.ylabel("Pressure (hPa)")
    plt.clabel(
        contour,
        inline=1,
        fontsize=5,
        fmt='%1.4f',
    )

    plt.xticks(np.linspace(15, 60, 10))
    plt.yscale('log')
    plt.yticks(np.linspace(1000, 100, 10))

    plt.gca().yaxis.set_major_formatter(
        plt.FormatStrFormatter('%.0f')
    )
    plt.colorbar(
        contour,
        orientation='vertical',
        shrink=0.7,
        label="(m/s)",
    )
    plt.gca().invert_yaxis()
    plt.ylim(1010, 99.99)
    plt.grid(
        visible=True,
        linestyle='--',
        alpha=0.5,
    )
    plt.savefig("120E_vetical_velocity.png")

    # plt.show()
```