標題: 不同緯度下地轉平衡平行與垂直等壓線距離、平行等壓線風速之探討

摘要:

本文將探討在給定條件的地轉平衡下，在不同緯度達到地轉平衡時所需之距離，以及在達地轉平衡後，平行等壓線方向的風速。在中學教育過程中，地轉平衡僅簡介在假定情況下的一般運動形式，並未提出緯度對平衡之距離的影響。而此可研究高空(可不計摩擦力)的情況下，氣塊運動過程牽涉的尺度，以及在各距離下地轉平衡的完整度。

因此本文將透過自行建立物理模型，並以程式模擬的方式，計算不同緯度下地轉平衡過程中平行與垂直等壓線距離的變化，並盡可能求取相關的數學式。

關鍵字: 地轉平衡、科氏效應、地轉風

研究動機:

因在研讀大氣相關書籍，關於地轉平衡與科氏效應部分章節時，並未提及探討關於緯度對地轉平衡所需之距離，因此希望透過此項模擬，找出緯度及平行與垂直等壓線距離的變化關係，以及平行等壓線方向風速與緯度關係。

研究方法:

本文內的內容透過物理建模、程式模擬為主軸進行探究

1. 物理建模

將假想氣塊視為質點，進行科氏力、氣壓梯度力共同作用下的物理參數關係式推導。

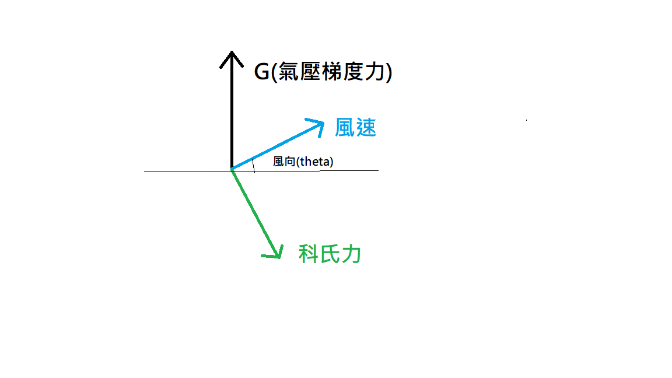
此物理建模基於給定之物理條件，與現實情況會有些許誤差。

1. 程式模擬:

利用物理建模所得之關係式，以數值解之方式模擬地轉平衡過程中氣塊的運動方式，待確認地轉平衡過程中，氣塊運動之軌跡無誤後，進行風速、風向兩物理量對時間變化之分析。

物理模型設計:

假設只有單向(上的氣壓梯度力，並給定相關物理常數，利用靜力平衡的概念，解出在氣壓梯度力與科氏力共同作用下，氣塊所受的加速度方程式。



待解出公式後，利用遞迴關係的概念撰寫迴圈，計算氣塊在各時間段中的物理性質(位置、速度、加速度)。並利用位置資料，撰寫繪圖指令。

由氣壓梯度力定義:

但由於模擬中只牽涉單一方縣的氣壓變化，因此可將上述公式改寫為:

由科氏力定義:

其中

根據上圖，可以得出下列兩方程式:

利用此兩數學式，帶入迴圈進行遞迴計算，模擬出氣塊的運動軌跡

在模擬過程中，假定氣塊在固定時間間距內進行等速度運動。

氣塊速度則以等加速度模擬。

程式模擬內容:

第一部分 – 計算數值

主程式:

1. # import function
2. import IoAgeo as I
3. # input the latitude
4. phi = input('Input the latitude:')
5. phi = float(phi)
6. # conduct the program
7. I.geostrophic(phi)

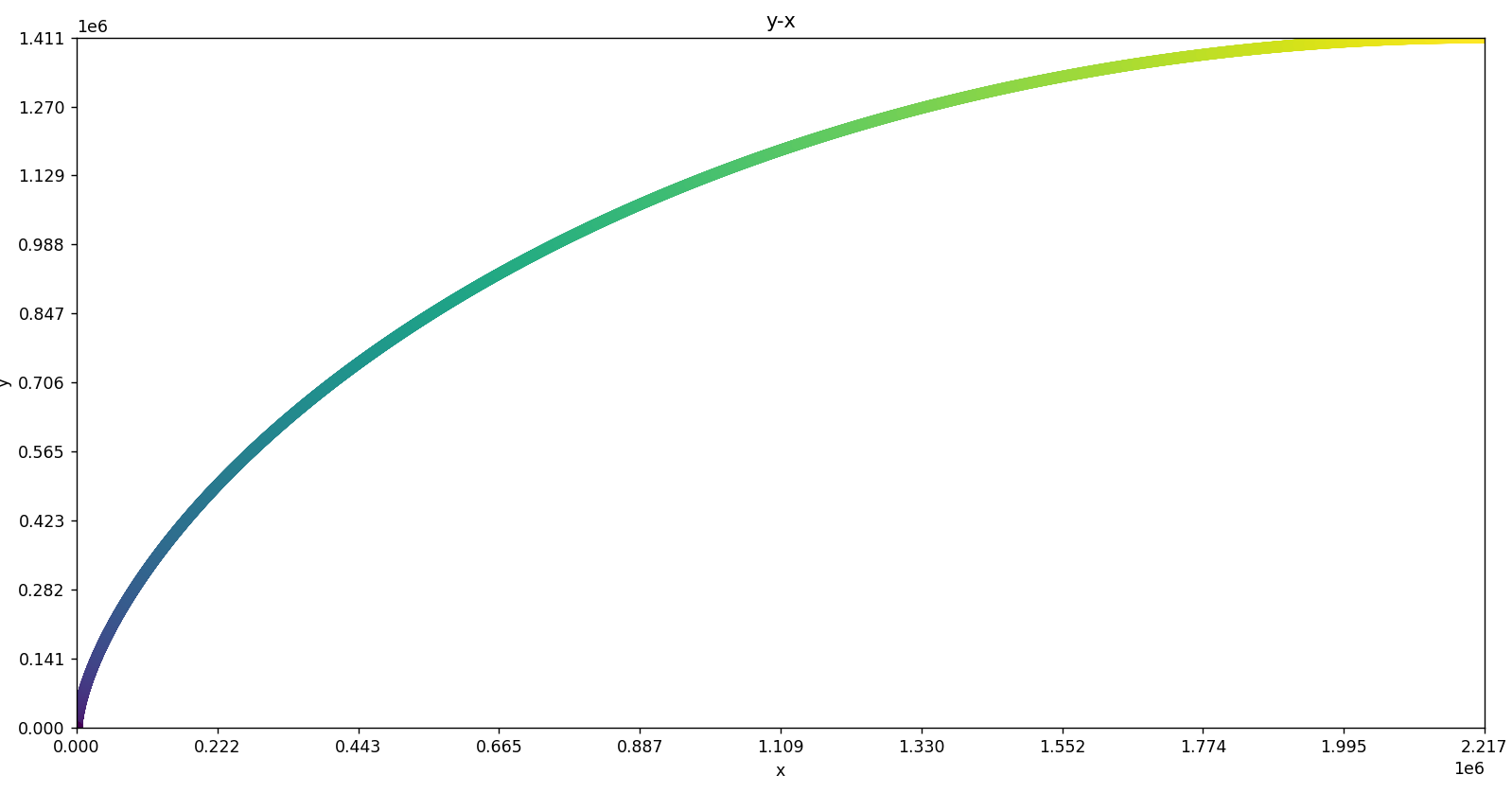
副程式:

1. def geostrophic(phi):
2. # import lib
3. import numpy as np
4. import math as m
5. import matplotlib.pyplot as plt
6. # setting constant
7. rho = 1
8. omega = 2\*(m.pi)/86400 # period of self rotating
9. phi = m.radians(phi) # assume the latitude is 45 degree\*
10. dt = 10
11. # setting array of changing variables
12. x = np.zeros(10000) # array of x-coordinate
13. y = np.zeros(10000) # array of y-coordinate
14. vx = np.zeros(10000) # array of x velocity
15. vy = np.zeros(10000) # array of y velocity
16. ax = np.zeros(10000) # array of x acceleration
17. ay = np.zeros(10000) # array of y acceleration
18. t = np.zeros(10000) # array of time
19. theta = np.zeros(10000) # array of degree of wind
20. P = np.zeros(10000) # array of pressure
21. # add nan
22. # setting initial condition of variables
23. x[0] = 0
24. y[0] = 0
25. vx[0] = 0
26. vy[0] = 0 # im m/s
27. #ax[0] = (-1/rho)\*(-1)+2\*omega\*vy[0]\*np.sin(phi)
28. #ay[0] = (-1/rho)\*(-1)
29. ax[0] = (-1/rho)\*(-1/1000)+2\*omega\*vy[0]\*np.sin(phi)
30. ay[0] = (-1/rho)\*(-1/1000)
31. t[0] = 0
32. theta[0] = np.radians(90)
33. # counting the data
34. for i in range(9999):
35. if vy[i]>=0:
36. t[i+1] = t[i]+dt
37. x[i+1] = x[i]+vx[i]\*dt
38. y[i+1] = y[i]+vy[i]\*dt
39. vx[i+1] = vx[i]+ax[i]\*dt
40. vy[i+1] = vy[i]+ay[i]\*dt
41. dp = -1/1000
42. Gy = (-1/rho)\*(dp)
43. ax[i+1] = 2\*omega\*vy[i+1]\*np.sin(phi)
44. ay[i+1] = Gy-2\*omega\*vx[i+1]\*np.sin(phi)
45. # print(i,round(x[i],2),round(y[i],2),round(vx[i],2),round(vy[i],2),round(ax[i],2),round(ay[i],2))
46. else :
47. break
48. # set maximum of position
49. xmax = np.max(x)
50. ymax = np.max(y)
51. print(round(xmax,5))
52. print(round(ymax,5))
53. # plot y-x diagram
54. plt.scatter(x,y,c=t)
55. plt.xticks(np.linspace(0,xmax+10,11))
56. plt.yticks(np.linspace(0,ymax+10,11))
57. plt.xlim([0,xmax+10])
58. plt.ylim([0,ymax+10])
59. plt.title('y-x')
60. plt.xlabel('x')
61. plt.ylabel('y')
62. plt.show()

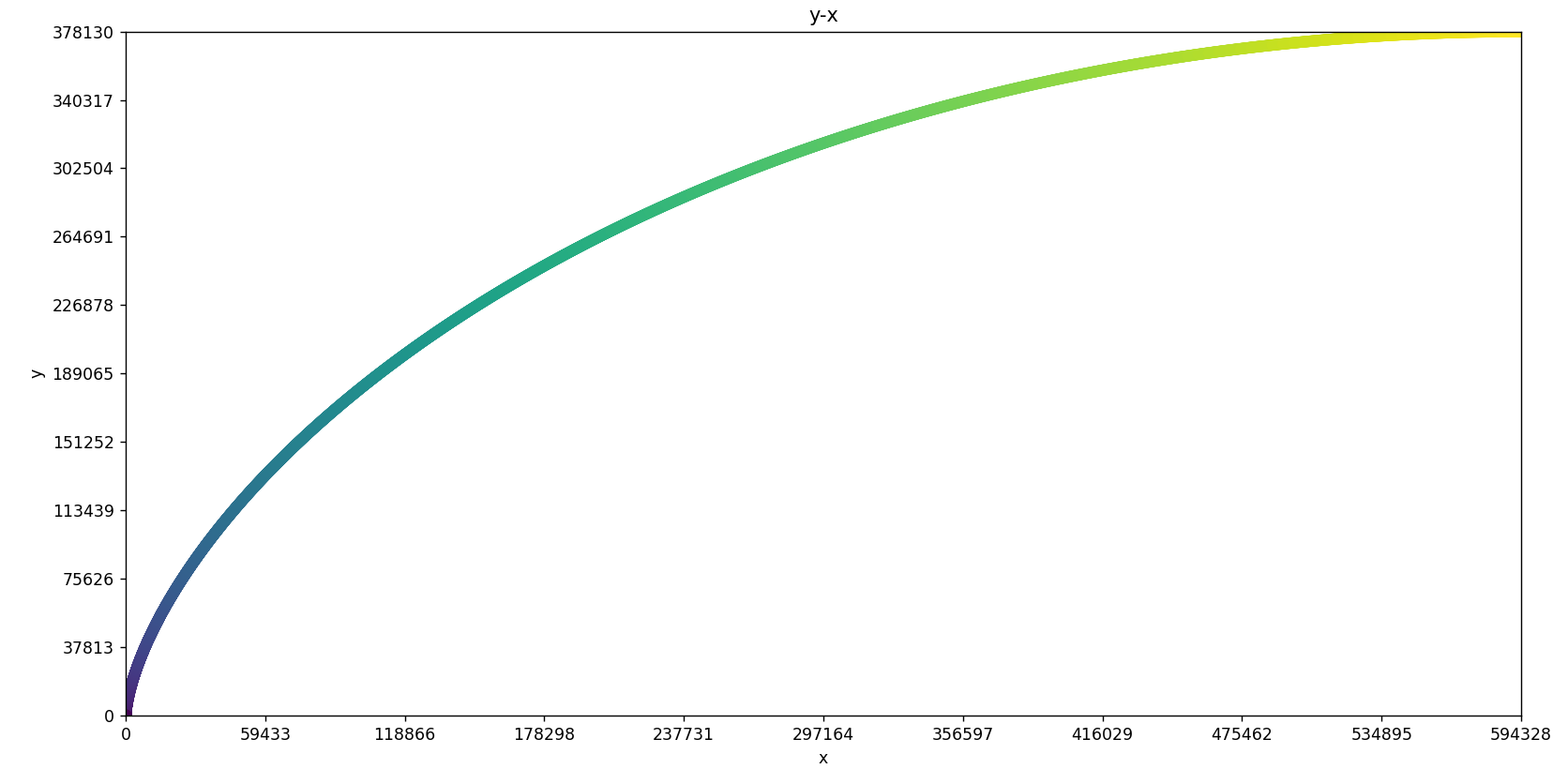
在此階段的執行結果除了數據之外，亦包含在使用者輸入緯度下，氣塊運動的軌跡圖形。

以下僅採部分緯度軌跡:

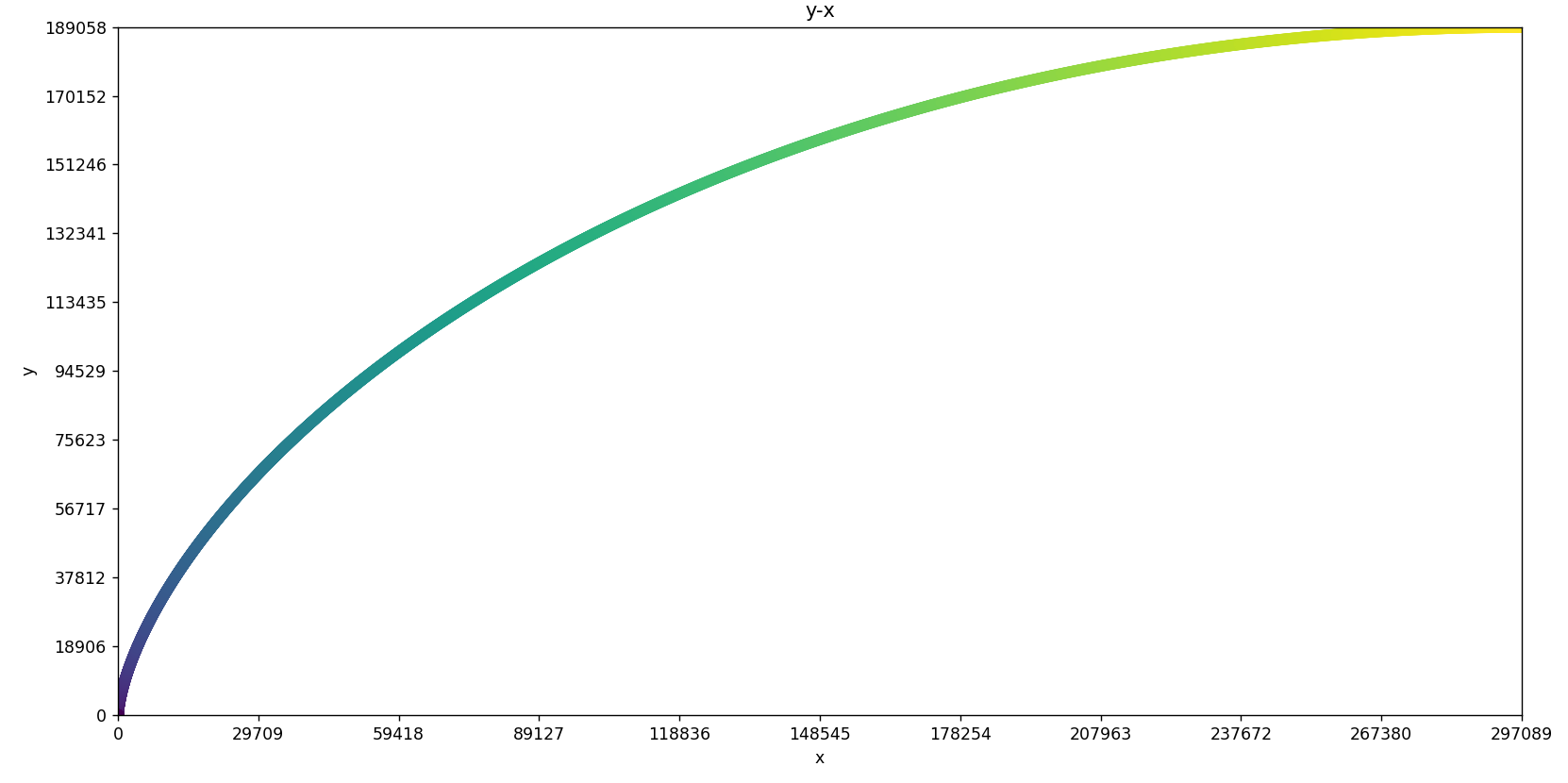
:



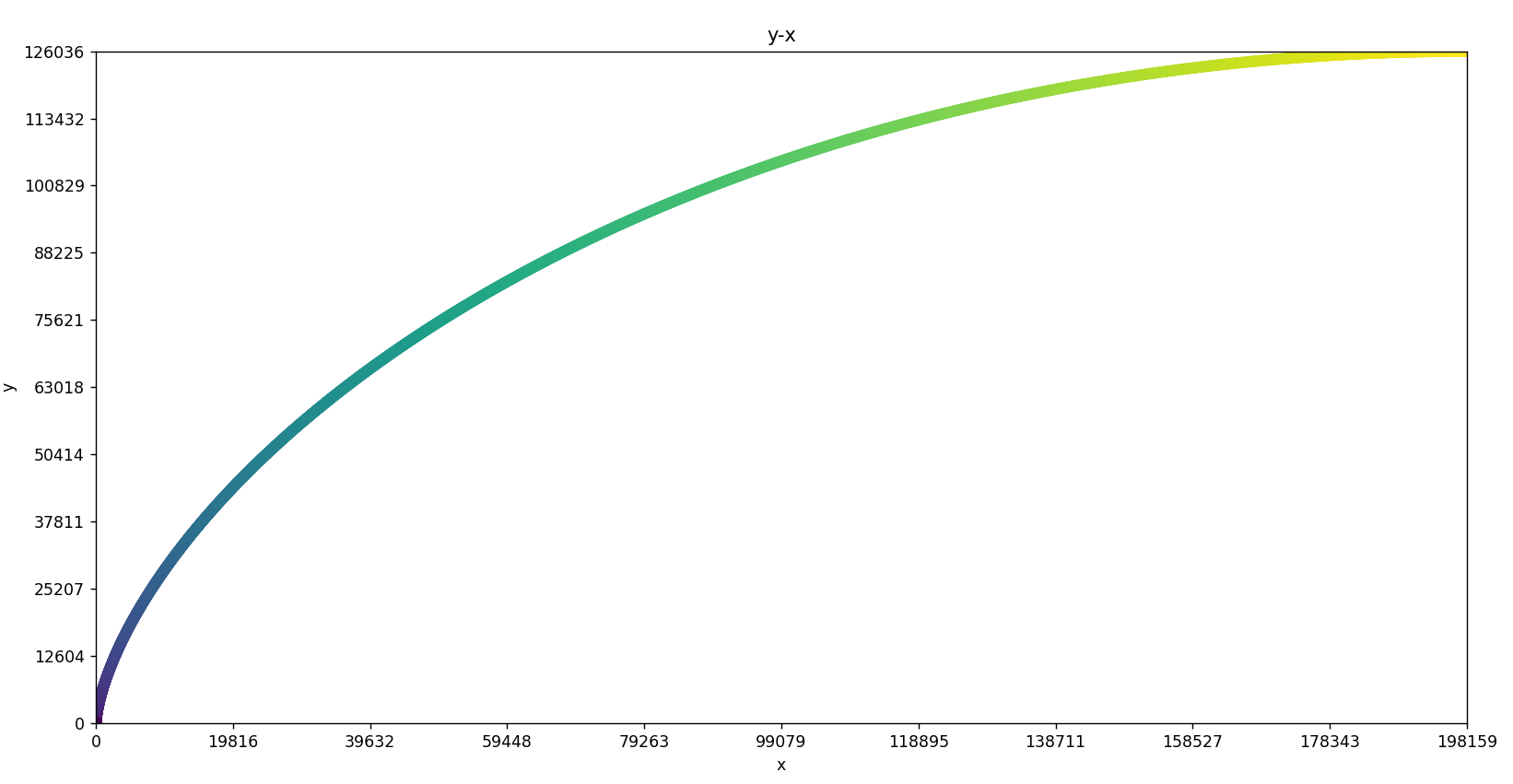
:



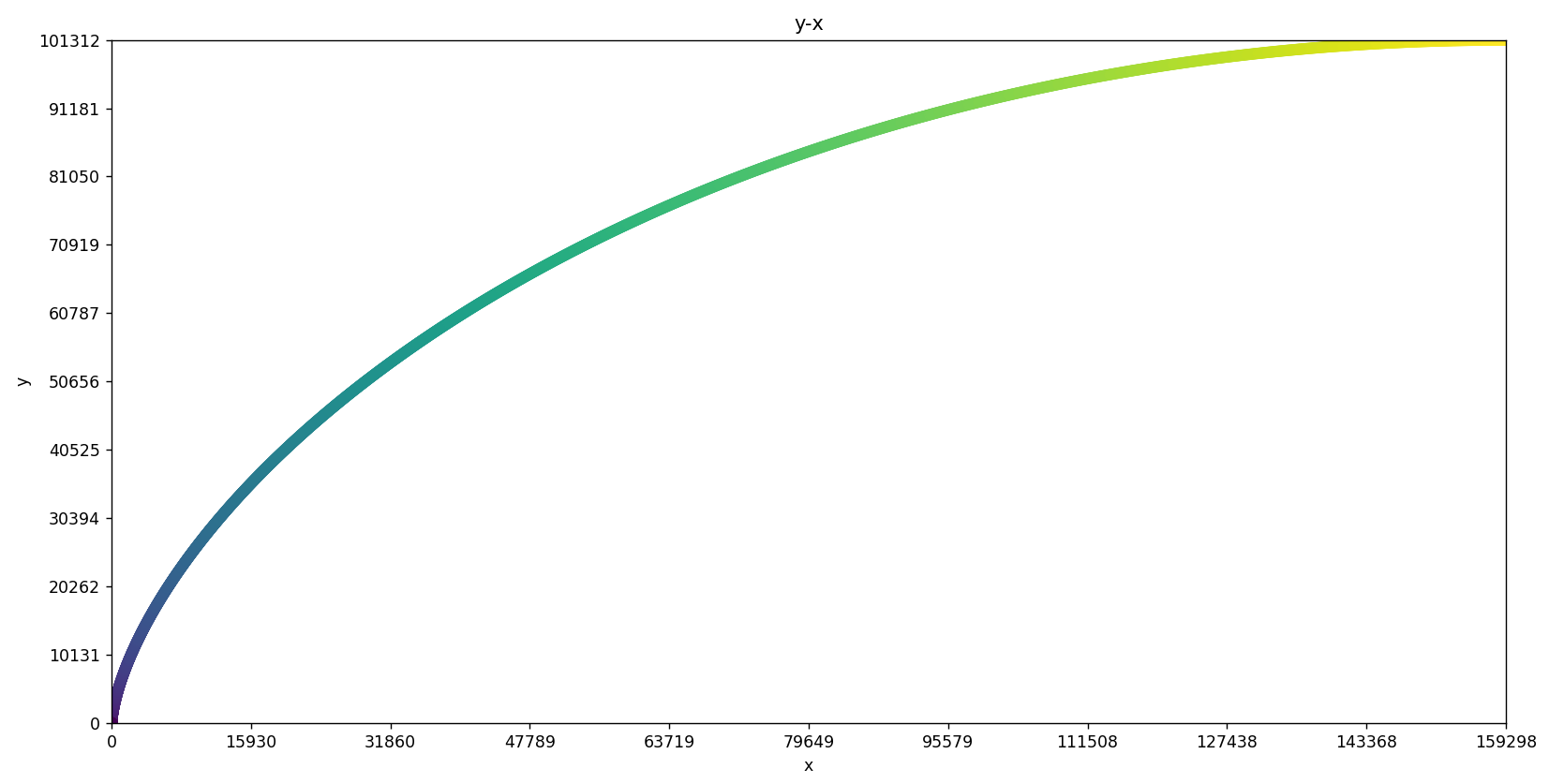
:



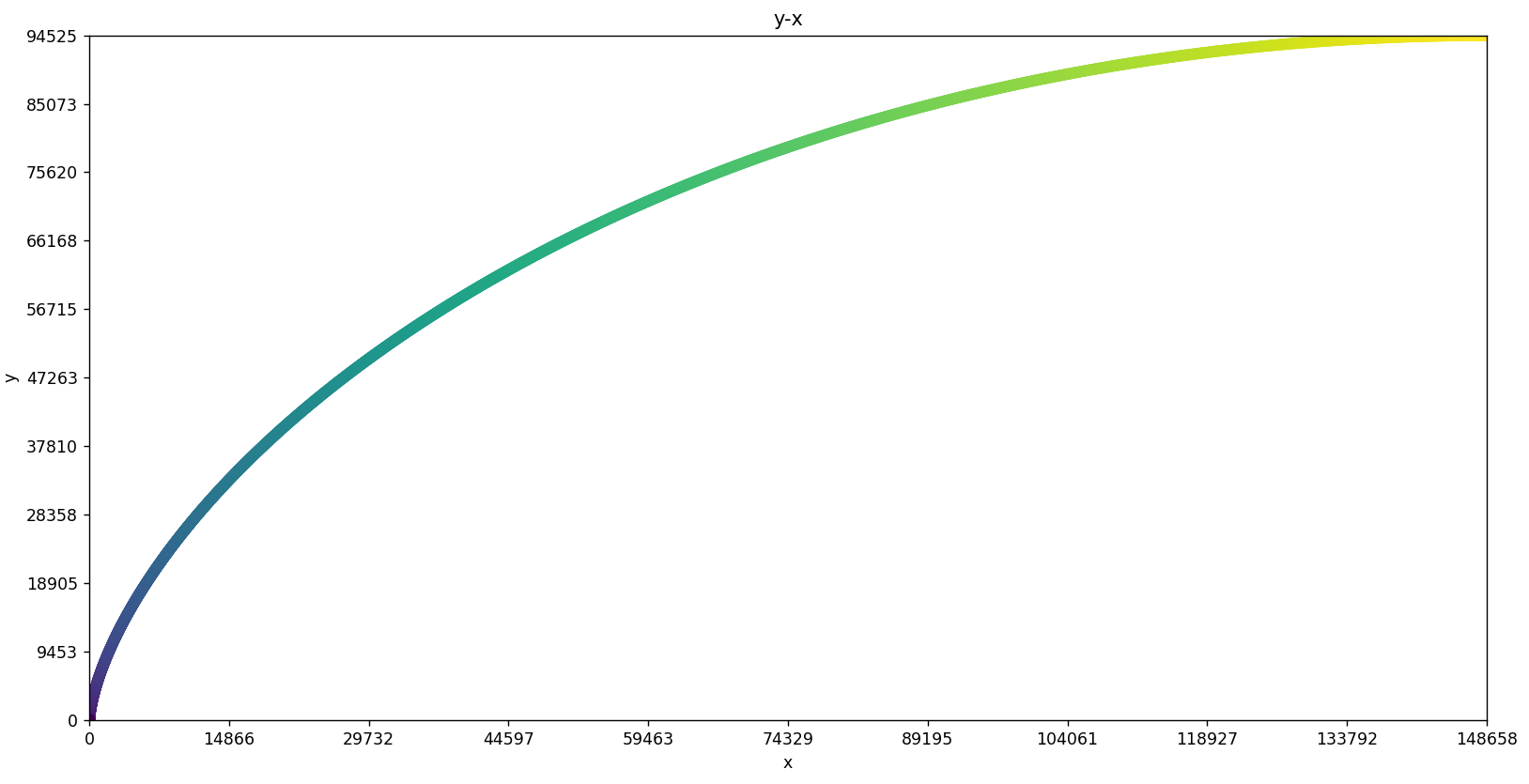
:



:



:



第二部分 – 繪製結果圖

主程式:

1. # import library
2. import numpy as np
3. import matplotlib.pyplot as plt
4. import test\_fig as t
5. # set array
6. x = np.zeros(90)
7. y = np.zeros(90)
8. vx = np.zeros(90)
9. lat = np.linspace(1,90,90)
10. # calculate x,y,u-wind
11. for i in range(1,91):
12. x[i-1],y[i-1],vx[i-1] = t.testgeof(i)
13. print(np.max(x))
14. print(np.max(y))
15. print(np.max(vx))
16. # draw figure
17. plt.plot(lat,x,'b:')
18. plt.title('The maximum distance (parallel to isobar) as latitude increase')
19. plt.xlabel('latitude\*10')
20. plt.ylabel('maximum distance (parallel to isobar)')
21. plt.xticks(np.linspace(0,90,7))
22. plt.yticks(np.linspace(0,3200000,6))
23. plt.xlim([0,90])
24. plt.ylim([0,3200000])
25. plt.show()
26. plt.plot(lat,y,'b:')
27. plt.title('The maximum distance (vertical to isobar) as latitude increase')
28. plt.xlabel('latitude\*10')
29. plt.ylabel('maximum distance (vertical to isobar)')
30. plt.xticks(np.linspace(0,90,7))
31. plt.yticks(np.linspace(0,5000000,11))
32. plt.xlim([0,90])
33. plt.ylim([0,5000000])
34. plt.show()
35. plt.plot(lat,vx,'b:')
36. plt.title('The maximum u-wind velocity as latitude increase')
37. plt.xlabel('latitude\*10')
38. plt.ylabel('maximum u-wind velocity')
39. plt.xticks(np.linspace(0,90,7))
40. plt.yticks(np.linspace(0,75,5))
41. plt.xlim([0,90])
42. plt.ylim([0,75])
43. plt.show()

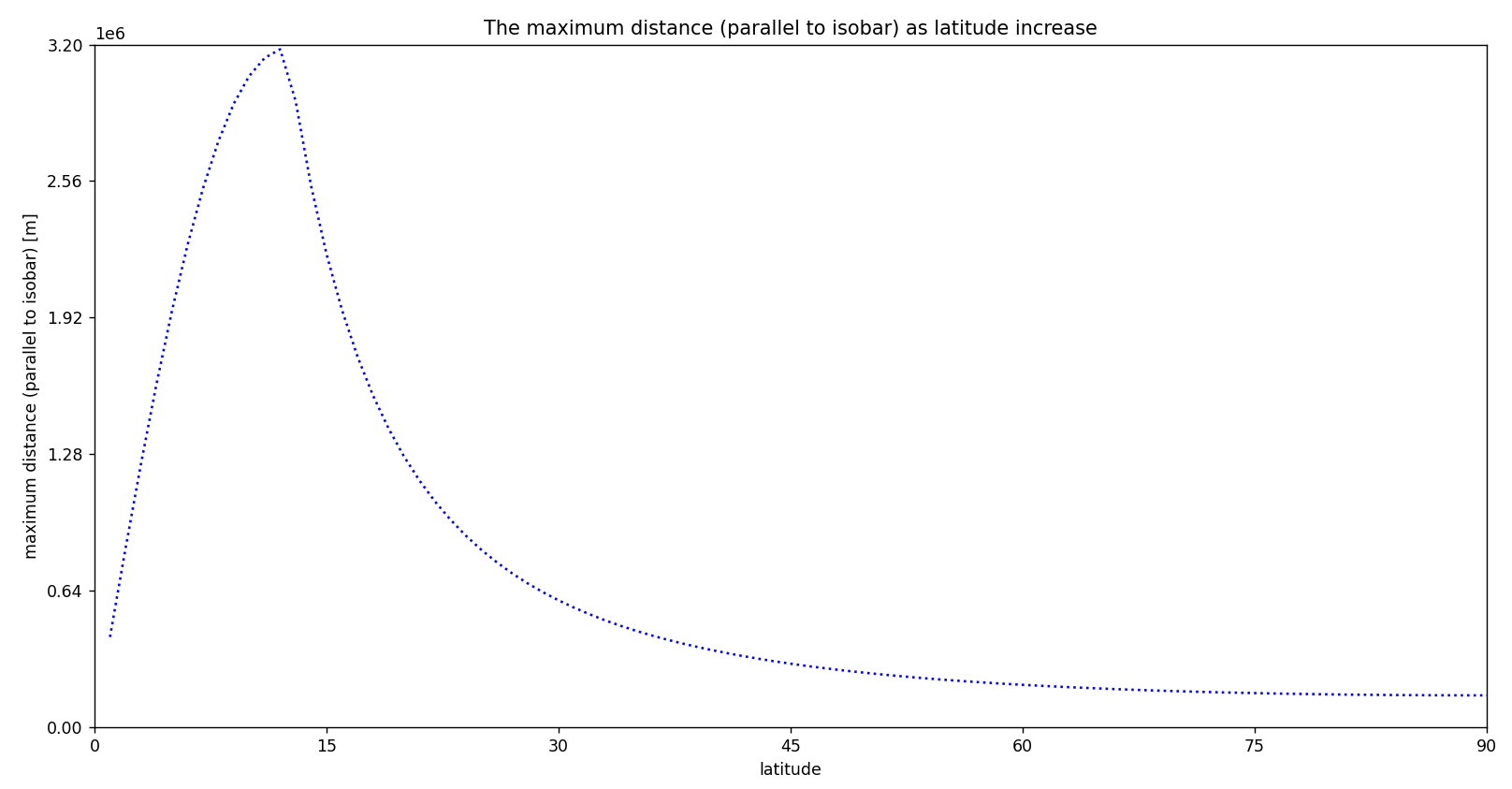
此主程式提供氣塊最大位移與最大水平速度隨緯度變化的關係圖

副程式:

1. def testgeof(phi):
2. # import lib
3. import numpy as np
4. import math as m
5. import matplotlib.pyplot as plt
6. # setting constant
7. rho = 1
8. omega = 2\*(m.pi)/86400 # period of self rotating
9. phi = m.radians(phi) # assume the latitude is 45 degree\*
10. dt = 10
11. # setting array of changing variables
12. x = np.zeros(10000) # array of x-coordinate
13. y = np.zeros(10000) # array of y-coordinate
14. vx = np.zeros(10000) # array of x velocity
15. vy = np.zeros(10000) # array of y velocity
16. ax = np.zeros(10000) # array of x acceleration
17. ay = np.zeros(10000) # array of y acceleration
18. t = np.zeros(10000) # array of time
19. theta = np.zeros(10000) # array of degree of wind
20. P = np.zeros(10000) # array of pressure
21. # setting initial condition of variables
22. x[0] = 0
23. y[0] = 0
24. vx[0] = 0
25. vy[0] = 0 # im m/s
26. ax[0] = (-1/rho)\*(-1/1000)+2\*omega\*vy[0]\*np.sin(phi)
27. ay[0] = (-1/rho)\*(-1/1000)
28. t[0] = 0
29. theta[0] = np.radians(90)
30. # counting the data
31. for i in range(9999):
32. if vy[i]>=0:
33. t[i+1] = t[i]+dt
34. x[i+1] = x[i]+vx[i]\*dt
35. y[i+1] = y[i]+vy[i]\*dt
36. vx[i+1] = vx[i]+ax[i]\*dt
37. vy[i+1] = vy[i]+ay[i]\*dt
38. dp = -1/1000
39. Gy = (-1/rho)\*(dp)
40. ax[i+1] = 2\*omega\*vy[i+1]\*np.sin(phi)
41. ay[i+1] = Gy-2\*omega\*vx[i+1]\*np.sin(phi)
42. # print(i,round(x[i],2),round(y[i],2),round(vx[i],2),round(vy[i],2),round(ax[i],2),round(ay[i],2))
43. else :
44. break
45. # set maximum of position
46. xmax = np.max(x)
47. ymax = np.max(y)
48. vxmax = np.max(vx)
49. return(xmax,ymax,vxmax)

研究成果:

藉由第二段程式可以得到下列三張成果圖:



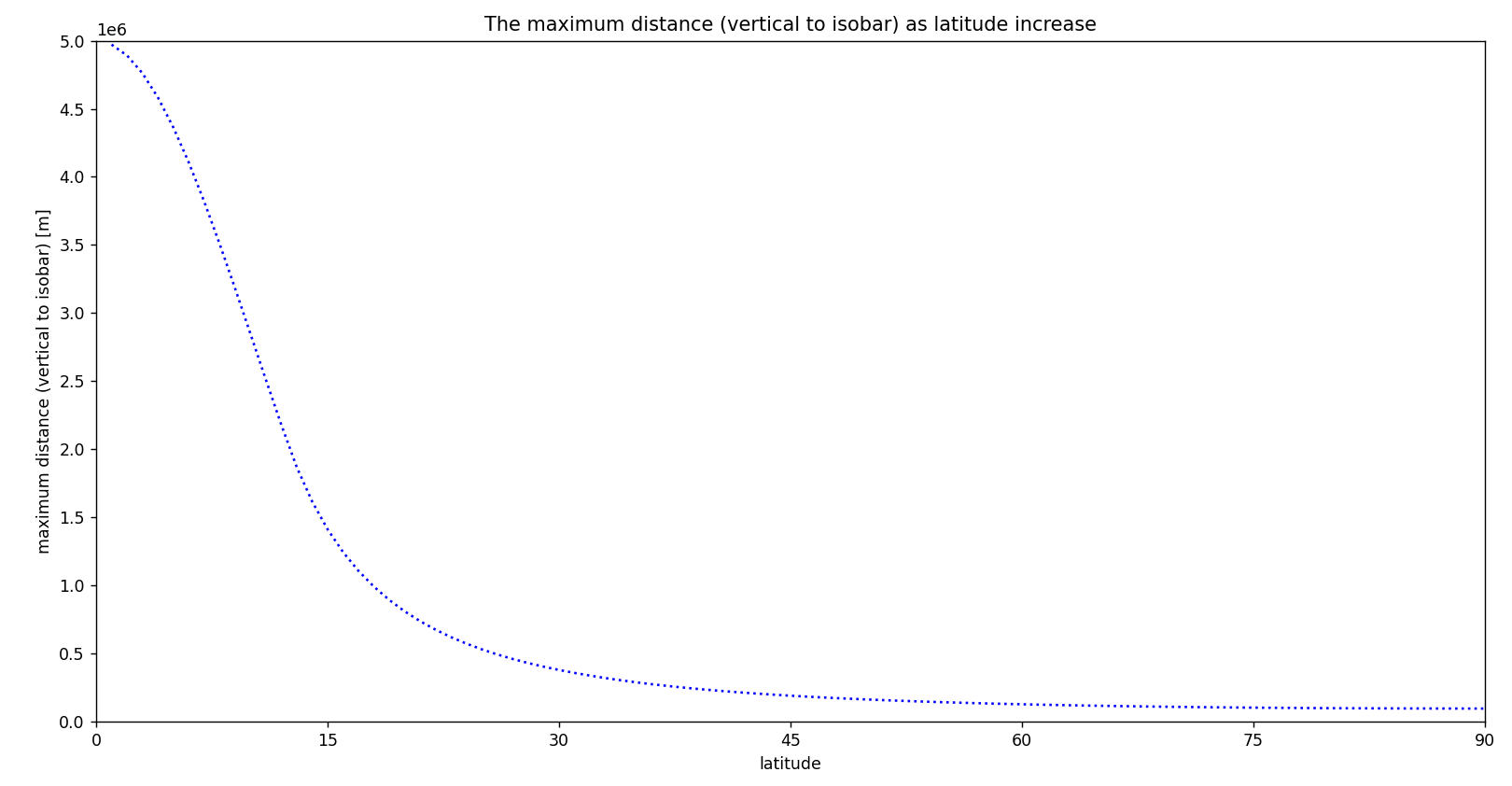
此張圖片為平行於等壓線方向最大位移，隨緯度變化的結果，其中可知，最大值約為，此時緯度為北緯。

在高緯地區，因科氏力較強，可以較快達到地轉平衡，因此只需較短距離即可達成地轉平衡。

而最大值位於北緯，而不是位於接近，其可能原因在於當緯度過低時，物體進行垂直等壓線方向距離較長，導致時間延長，而氣塊的氣壓梯度力則須在較中後段才出現顯著影響，因此其水平距離並不會是最長。

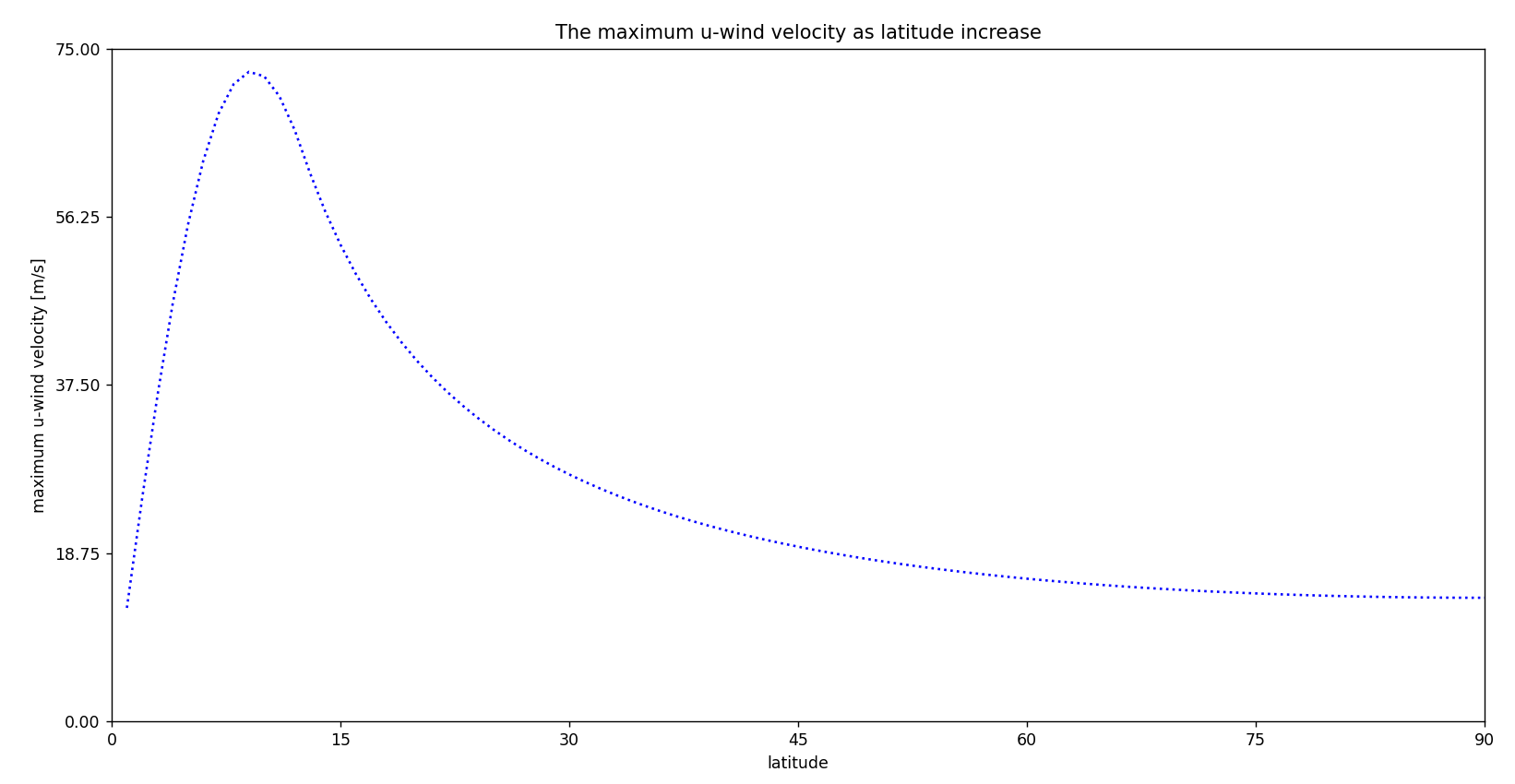
經使用Excel 對數趨勢線，得出在前，方程式為:

在之後，方程式為



此圖為垂於等壓線方向最大位移隨緯度變化圖。

很明顯在低緯及高緯地區有顯著的差異，隨著緯度漸增，垂直位移減少的速度漸減，可見約在前，垂直等壓線方向的位移變化相當劇烈。



此圖為氣塊最大速度隨緯度變化的情況，

在氣塊最大速度約在左右達到最大值。

在低緯地區因平行等壓線方向的作用較弱，無法提供較大的作用。

而在高緯地區因為較快達到地轉平衡，因此達平衡時的速度並不會是最大。

探討結果應用:

因此物理模型並未考慮摩擦力以及其他實際大氣中的物理現象，因此模擬的結果並非絕對準確。若在此模型中加入摩擦力、紊流等大氣中常見的物理特性以及影響氣塊運動的因子，就可以更加精確的模擬出實際情況。若達成此項目標，可以對於在不同緯度之間的氣塊運動模式有更深的了解，可進一步依該模型研究整體的大氣環流模式，或者特定緯度高空風場的形式。

參考資料:

1. 氣壓梯度力-維基百科

<https://zh.wikipedia.org/zh-tw/%E5%A3%93%E5%BC%B7%E6%A2%AF%E5%BA%A6%E5%8A%9B>

1. 科里力奧力-維基百科

<https://zh.wikipedia.org/zh-tw/%E7%A7%91%E9%87%8C%E5%A5%A5%E5%88%A9%E5%8A%9B>