涡度方程数值实验

姜文瑞 - 17307110078

此次作业首先用fortran实现了最简单的欧拉显式差分格式，调用csv_file模块输出为csv文件，并用python进行画图；整个过程用make全自动进行。之后的每次模拟都只在这个最简单的模拟的基础上做**一处**改动。我们首先检验了不同的时间分辨率造成的影响。之后我们改变κ, f_0, β的值，对这三个参数进行了比较初步的敏感性分析。最后选用其他的时间积分格式，包括Leap-frog格式，松野格式，对时间积分格式的性能也做了一些分析。

向前差显式格式（对照组）

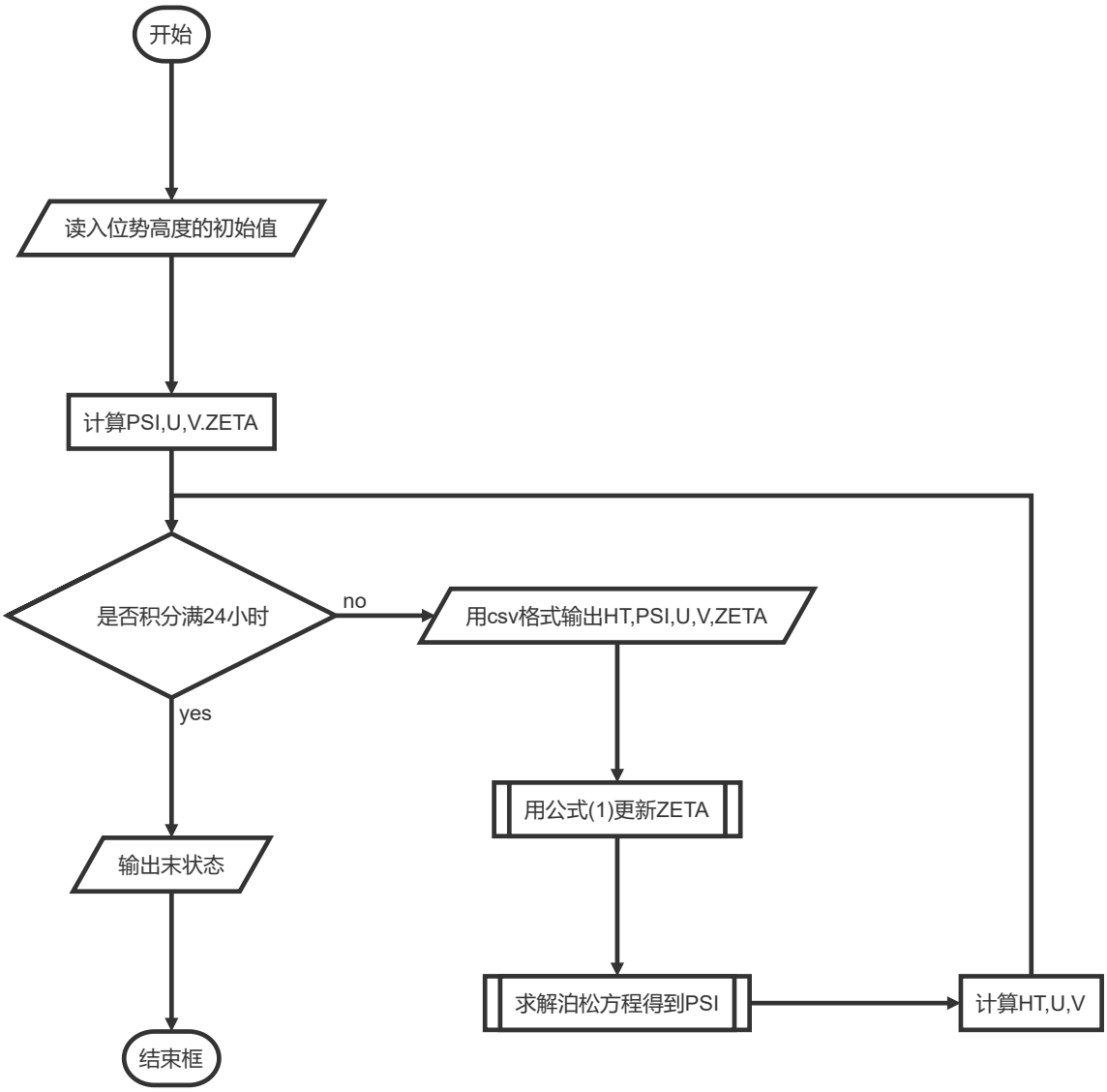
向前差显格式的核心步骤是：

$$\frac{\zeta^{n+1}-\zeta^n}{\tau}=-u^n\frac{d_h\zeta^n}{d_hx}-v^n\frac{d_h\zeta^n}{d_hy}+\kappa\nabla_h^2\zeta^n \tag{1}$$

其中τ为时间分辨率，ζ^n,u^n,v^n为$t=t_n$时刻的涡度、纬向速度和经向速度，κ为涡度扩散系数，$\frac{d_h}{d_hx},\frac{d_h}{d_hx},\nabla_h$为离散的空间梯度（在边界上取单边差分，在内部取中央差分）。这里时间步长选取一小时，也就是3600秒。后面的实验都只在此实验基础上做一处改动。

流程图

Fortran执行的动力学部分的流程图如下：

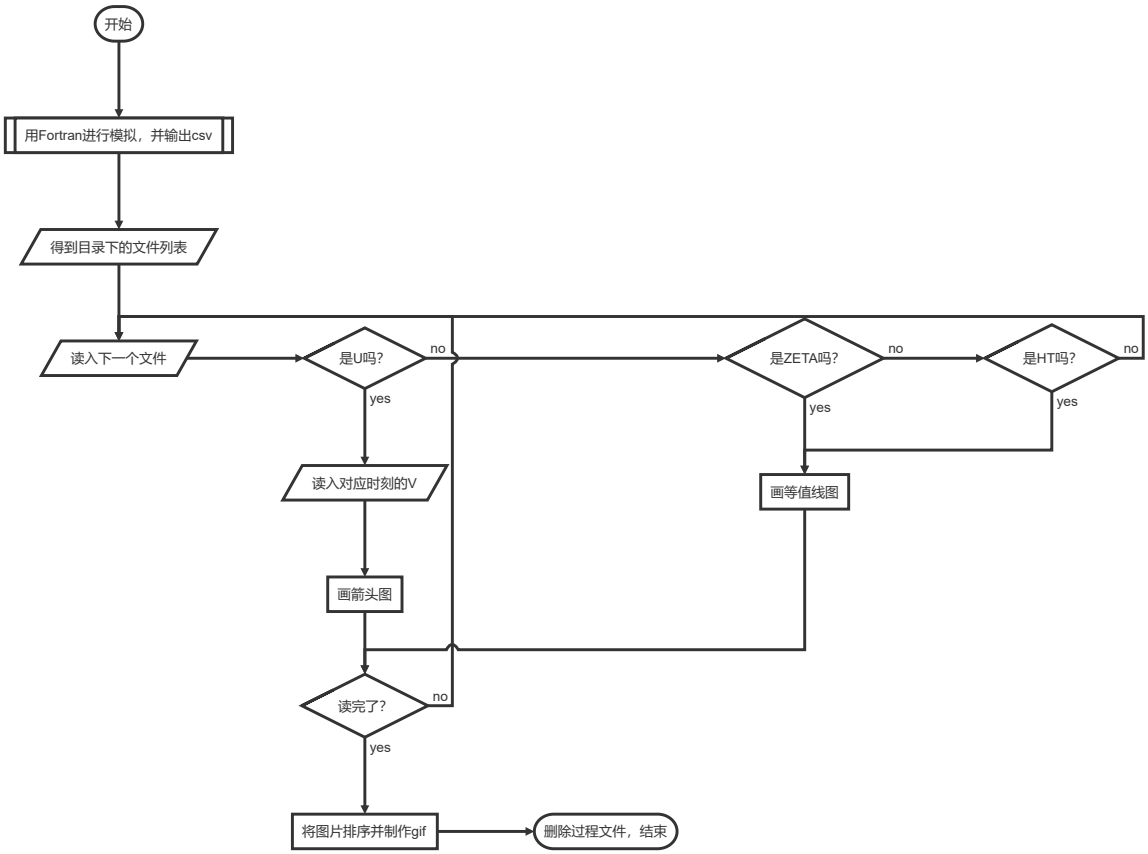


因为Fortran没有画图模块，在实际操作中我使用了make，达到了类似python, fortran, shell 三种语言混编的效果。Makefile的内容如下：

```
flow.gif ht.gif: mass_plot.py u%.csv v%.csv ht%.csv
    /home/mace/c/Anaconda3/python.exe mass_plot.py
    rm *.csv
    rm flow*.png
    rm dvor*.png
    rm vor*.png
    rm ht*.png

u%.csv v%.csv ht%.csv: mine.f90
    gfortran -o mine mine.f90 csv_file.o
    ./mine
```

它所实现的整个运行和画图流程如下：



常数的选取

f_0和β的选取

这两个常数的表达式分别为

$$f_o = 2\Omega \sin\varphi$$
$$\beta = \frac{df_0}{dy} = \frac{d\varphi}{dy}\frac{df_0}{d\varphi} = \frac{2\Omega\cos\varphi}{R}$$

期中φ, Ω, R分别代表纬度，地球自转频率和地球半径。模拟的中心纬度$45^\circ N$作为φ，取$\Omega = 7.2921 \times 10^{-5} s^{-1}, R = 6.371 \times 10^6 m$。求得

$$f_0 = 1.031 \times 10^{-4}$$
$$\beta = 1.619 \times 10^{-11}$$

κ的选取

涡度的扩散系数在实验上是一个非常难以计算的数字，为了选取合适的κ值，我参考了[1]。文中提到的多个模型κ从7×10^4至2×10^6不等。作为基础款的模拟，我们选用了$\kappa = 0$。这一方面是为了减少主观因素，另一方面也是为了使得对照组可以更好的凸显出平滑和其他积分格式的作用。κ的其他取值将在敏感性分析中给出。

周期性边界条件的泊松方程解法

解法的核心思想是用快速傅里叶变换后的乘法代替求导/积分运算。这一部分要解决的问题是：

$$\Delta\psi = \zeta$$

ψ在y方向上取固定的值，在x方向上取周期性边界条件。如果我们先不考虑边界条件，并且假设导数在无穷远处趋于0，在等号两边同时做傅里叶变换，并分部积分：

$$F[(\partial_x^2 + \partial_y^2)\psi] = F[\zeta]$$
$$\Rightarrow [(ik_x)^2 + (ik_y)^2]F[\psi] = F[\zeta]$$

因此

$$\psi = F^{-1}\left[\frac{-F[\zeta]}{k_x^2 + k_y^2}\right]$$

解的主干大致如此，在实际操作的时候程序首先在y方向上复制了一个一模一样的矩阵，使得在y方向上ζ变成了奇函数，方便后续提取有用的因子。因为傅里叶变换输出的波数依次是：0，1，2.....$k_c - 1$，$-k_c$，.....-1，（k_c为Nyquist临界波数）所以在处理x方向的数据时，同时取出的两个数据并不对应相同的波数，需要做一定的调整（其实我这段真的没看懂，但我猜测是干这个用的）。

最后我们还需要考虑函数的边界条件。由于左边的拉普拉斯算子是二次的，对于任意的a,b有：

$$\Delta(ay + b) = 0$$

由此可以满足y方向上的固定边界条件。在x方向上则直接将西边界上的值设定成东边界上的值即可。

FFT算法原理

程序中调用的是Numerical Recipe的多维FFT算法(正向逆向各一次)，程序读入n维数组变形而成的一维数组，相邻的两位代表复数的实部和虚部。普通的FFT算法可以通过下面的式子简单的拓展到高维（以二维为例）：

$$\begin{aligned} H(n_1, n_2) &= FFT_1(FFT_2[h(k_{,1}\, k_2)]) \\ &= FFT_2(FFT_1[h(k_{,1}\, k_2)]) \end{aligned}$$

其中FFT_1, FFT_2分别代表对第1，2个坐标轴做一维的快速傅里叶变换。接下来我将介绍一下FFT的实现方法，以及程序中其他的细节。FFT算法的核心是Danielson-Lanczos引理：

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{2\pi i(2j)k/N} f_j + \sum_{j=0}^{N/2-1} e^{2\pi i(2j+1)k/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_j + W^k \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_j \\
&= F_k^{\text{偶}} + W^k F_k^{\text{奇}}
\end{aligned}
$$

其中$W = e^{2\pi i/N}$。每一次操作都将要完成的任务分成了两份，只考虑奇数或只考虑偶数。这个过程可以重复的进行1分2，2分4将矩阵逐步细分。程序要求N是2的整次幂，因此最终分到的每个部分就只有一个数，此时FFT的计算就没有难度了。

$$
F_k^{\text{奇 偶 偶 } \cdots \text{ 奇 偶}} = f_j
$$

如果将上标倒序，将每个“偶”变成0，将每个“奇”变成1我们就得到了j的二进制表示。这也就是为什么在程序的一开始要将数组颠倒一下。

除此之外另一个有趣的地方是三角函数值的计算，这个算法在写的时候非常细腻，为了避免重复计算三角函数值（比较昂贵），使用了递推的方法：

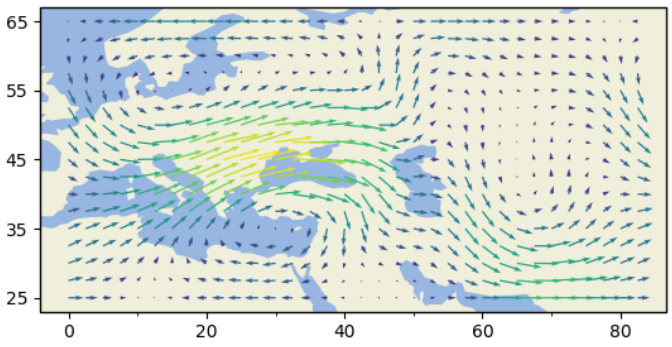
$$
\begin{aligned}
\cos(\theta + \delta) &= \cos\theta - [a\cos\theta + b\sin\theta] \\
\sin(\theta + \delta) &= \sin\theta - [a\sin\theta - b\cos\theta]
\end{aligned}
$$

其中$a = 2\sin^2\left(\frac{\delta}{2}\right), b = \sin\delta$，$\delta$为每次增加的角度。虽然这个公式很简单，可以用和角公式快速得到，但确实很大程度上加快了计算。

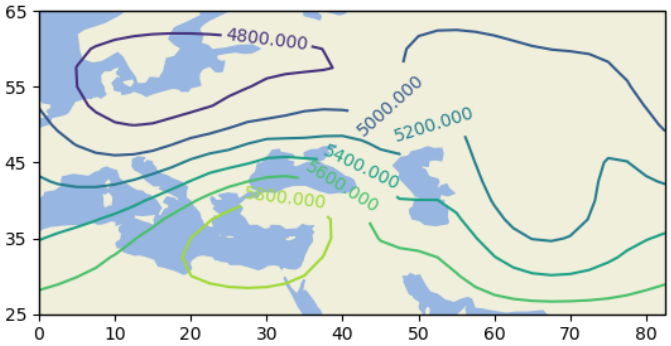
模拟结果

24小时预报结果如下：

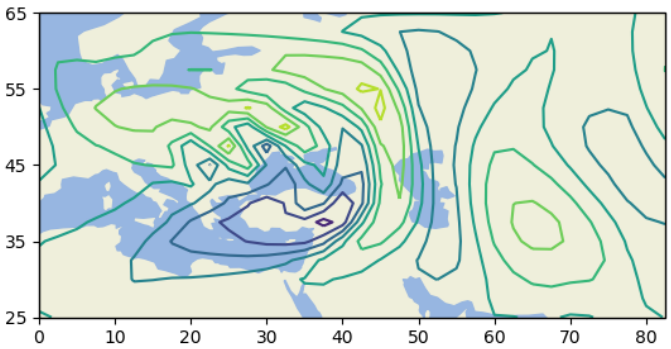
风场：



位势高度：



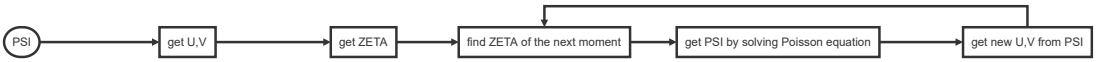
涡度场：



虽然流场和高度场还比较光滑，但从涡度场已经可以看到一些高频的信号被明显的放大了。这说明向前差格式基本可以胜任24小时预报，但是还是有进步的空间。

PSOLVE和差分引入的误差

起初在实现向前差格式的时候我犯了这样的一个错误，正常的流程本来应该是



我在设计的时候，为了使每个时刻的ZETA都对应当时的U,V，每次更新PSI的时候都重新更新一次ZETA，具体流程如下图。

而这导致了极大的误差，甚至淹没了整个物理过程。这背后的原因是由ZETA求解离散的泊松方程得到PSI和用PSI的中央差得到ZETA并不是彼此的逆运算。

在查阅资料的过程中我看到了另一种求解泊松方程的方法[2]，这个方法是在随机的三角形网格上的有限元方法。这个方法给了我一定的启发。如果把PSI变形成一个NX*NY维的列向量$\vec{\psi}$，那么求解ZETA的过程可以写成：

$$\vec{\zeta} = A\vec{\psi} \tag{2}$$

其中A代表内部中央差分，边界上单侧差分求得ZETA的算符，它可以表示成：

$$A = D_x \otimes I_y + I_x \otimes D_y \tag{3}$$

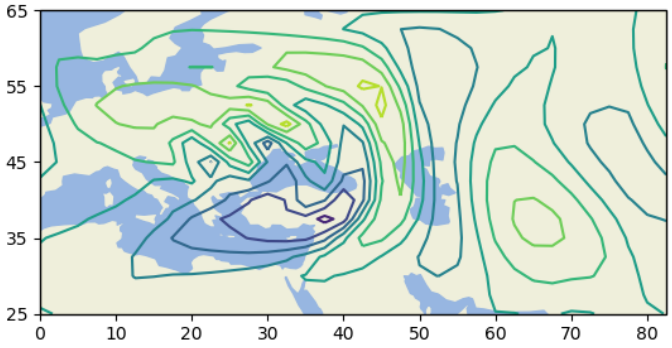
D_x, D_y, I_x, I_y分别为x,y方向上的一维微分矩阵和单位矩阵，不难证明矩阵A是对角占优的。已知$\vec{\zeta}$求解线性方程组（2）就可以达成PSOLVE的目的，此时就不需要担心U,V和ZETA不对应的问题了。

值得注意的是虽然求解A的逆需要的时间是$O(NX^3NY^3)$，但矩阵A并不随时间变化，因此求一次逆就可以一劳永逸，每次求解PSI只需要用$O(NX \cdot NY)$的时间计算一个内积即可。相较之下二维FFT算法每一次的时间复杂度是$O(NX\log(NX)NY\log(NY))$，因此在积分的步数比较多的时候，直接求解线性方程所用的时间是比FFT算法更快的。

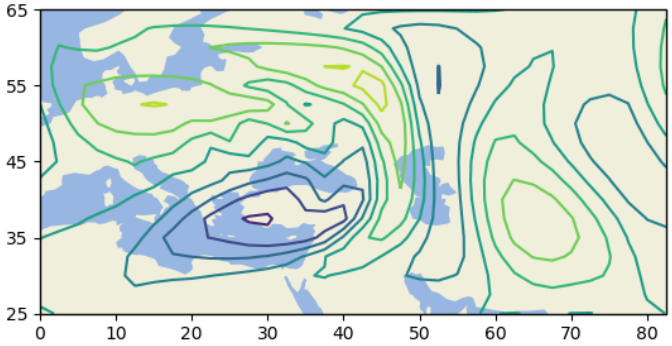
比较遗憾的是，我安装的gfortran的lapack包在链接的时候遇到了一些问题，因此没能自己实现这一想法。

改变时间分辨率

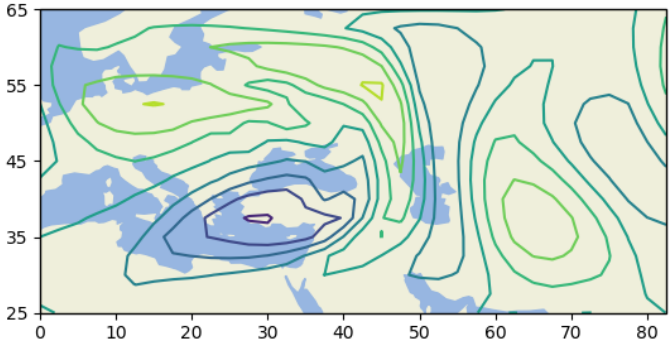
同为向前差显格式，选时间分辨率分别为3600s,1800s,900s，我们做了如下三次模拟： $\tau = 3600$



$\tau = 1800$



$\tau = 900$



容易看出时间分辨率的增加抑制了高频信号的增长，这和预期非常一致。

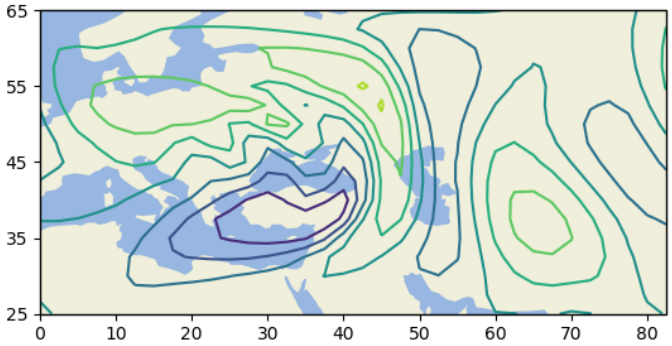
使用五点平滑

五点平滑的过程可以写为

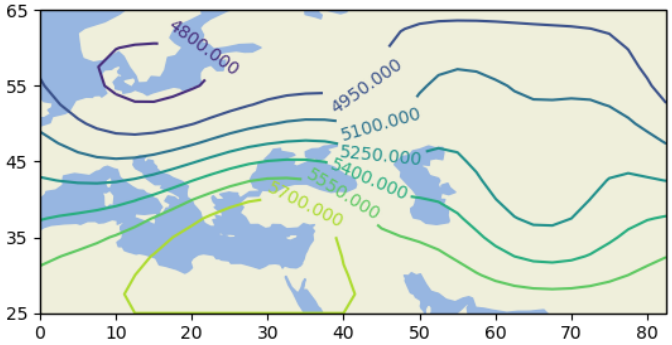
$$f_{i,j} = f_{i,j} - \frac{s}{4}(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j})$$

我们在实际操作时在每次更新过ZETA后，都对ZETA做一次5点平滑，这样理论上可以滤掉一部分高频波动。其中s是一个有一定主观色彩的参数，这个值直接决定了模拟过程中各种波长的信号随时间的衰减。参考了[1] 中s的选取，我采用了$s = 0.03, s = 0.1$这两种模拟。

在$s = 0.03$时，滤波有一定效果但并不完全令人满意：



而在取$s = 0.1$时，从位势高度上来看，有一部分有用的信息也被滤掉了：

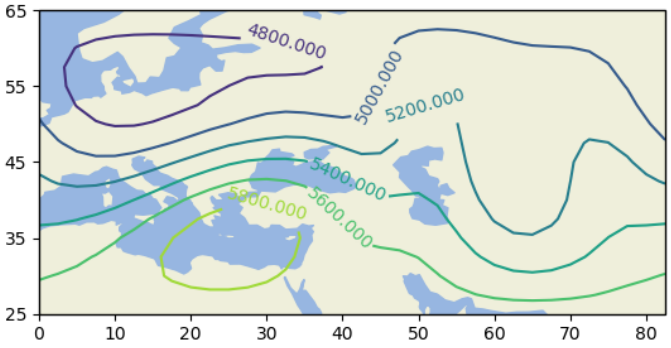


这说明用好平滑方法需要一些技术和经验，并不是想象中那么容易。

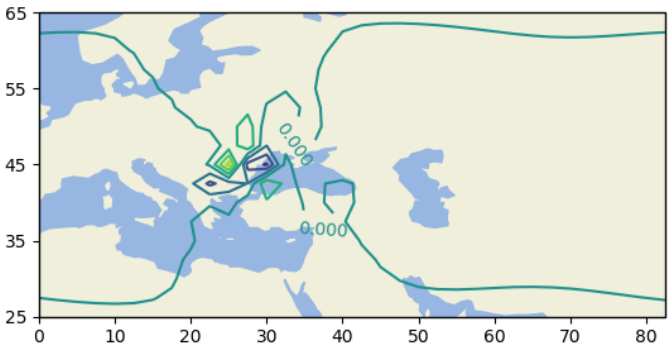
敏感性分析

f_0的敏感性分析

在所有的常数之中f_0是唯一一个没有在核心的时间积分步骤中出现的。它只出现在位势高度和流函数PSI的相互转换之中，理想情况下对PSI,ZETA,U,V的影响都是线性的，而对位势高度的预报没有影响。例如如果我们选取$65^\circ N$对应的地转频率，得到的位势高度预报几乎和没有改变f_0之前完全相同：



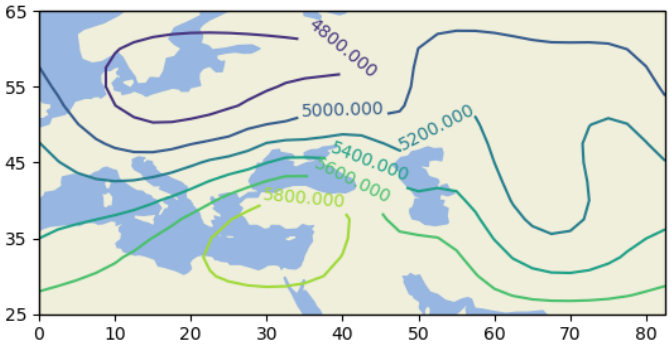
然而由于f_0较小是U,V较大，涡度方程的平流部分可能会发生不稳定，在选取$25^\circ N$对应的f_0带入模式时，确实有明显的不稳定出现：

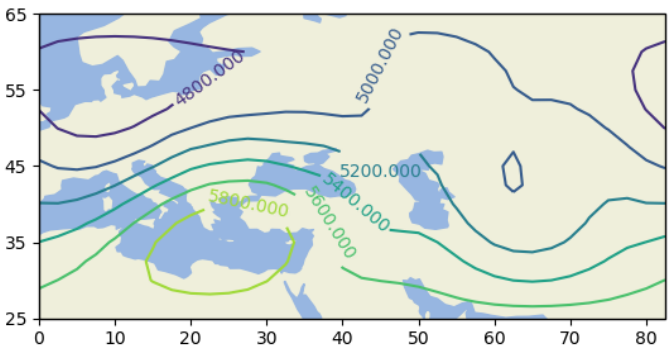


简而言之，模型在保证计算稳定性的情况下对f_0不敏感。

β的敏感性分析

从物理原理上来分析β可以影响Rossby波的速度，所以如果模型准确的话，预测结果就应该和β有关。我们对β为原本值的一半、为原本值的两倍分别进行了数值实验，结果依次为：



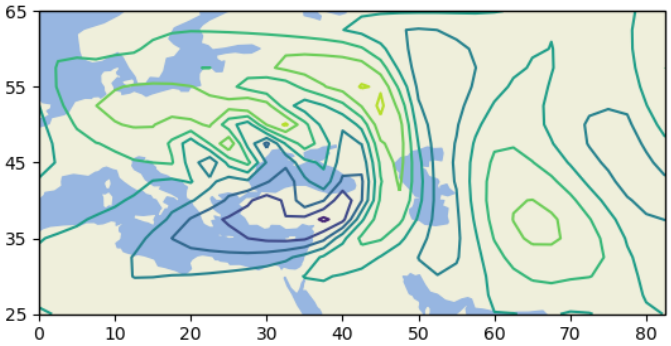


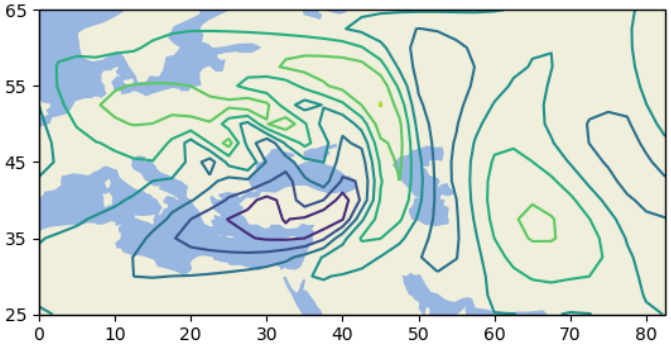
以上两组结果和对照组比较可以发现β越大高压向西移动越慢。这是因为β越大，高压东侧的北风因为β效应产生的正涡度越强，阻碍了高压向东移动。这个现象也可以用Rossby波的相速度理论来解释。

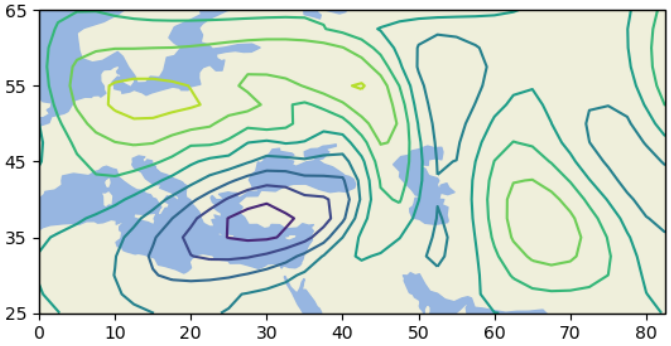
整体来看，模型比较好的捕捉到了β效应。模拟表明，β的大小对系统有一定的影响。但是在24小时内，即使β成倍的变化，对预报准确性的影响也不是很大。

κ的敏感性分析

这一系列实验中我们分别选取$\kappa = 7 \times 10^4, 2 \times 10^5, 2 \times 10^6$。最后一个量明显是偏大的，引入这一组实验主要是因为前两组的滤波结果并不十分理想。







虽然最后一组的滤波效果比较理想，但类似于平滑算法中$s = 0.1$那一组，这一组也丢失了一部分有用的信息。这说明κ的选取也是很需要知识和经验的。

总而言之，因为前两组得到的结果和对照组都十分接近，我们认为模型对κ并不十分敏感。

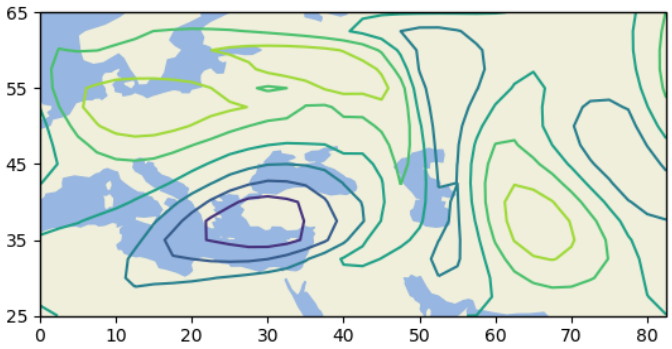
其他时间积分格式

松野格式

松野格式可以表示为

$$\begin{aligned}\frac{\zeta_*^{n+1}-\zeta^n}{\tau} &= -u^n\frac{d_h\zeta^n}{d_hx} - v^n\frac{d_h\zeta^n}{d_hy} + \kappa\nabla_h^2\zeta^n \\ \frac{\zeta^{n+1}-\zeta^n}{\tau} &= -u_*^n\frac{d_h\zeta_*^n}{d_hx} - v_*^n\frac{d_h\zeta_*^n}{d_hy} + \kappa\nabla_h^2\zeta_*^n\end{aligned} \tag{4}$$

不难看出松野格式直接使得计算量翻倍了，但也让格式更接近隐式格式了。流程图只需要更改时间积分这一处。最终预报的结果如下：



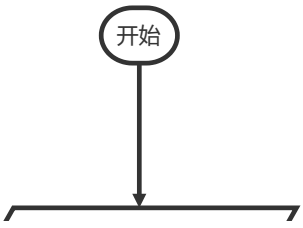
在24小时内没有高频信号的急剧增加，可以说模拟的挺不错。

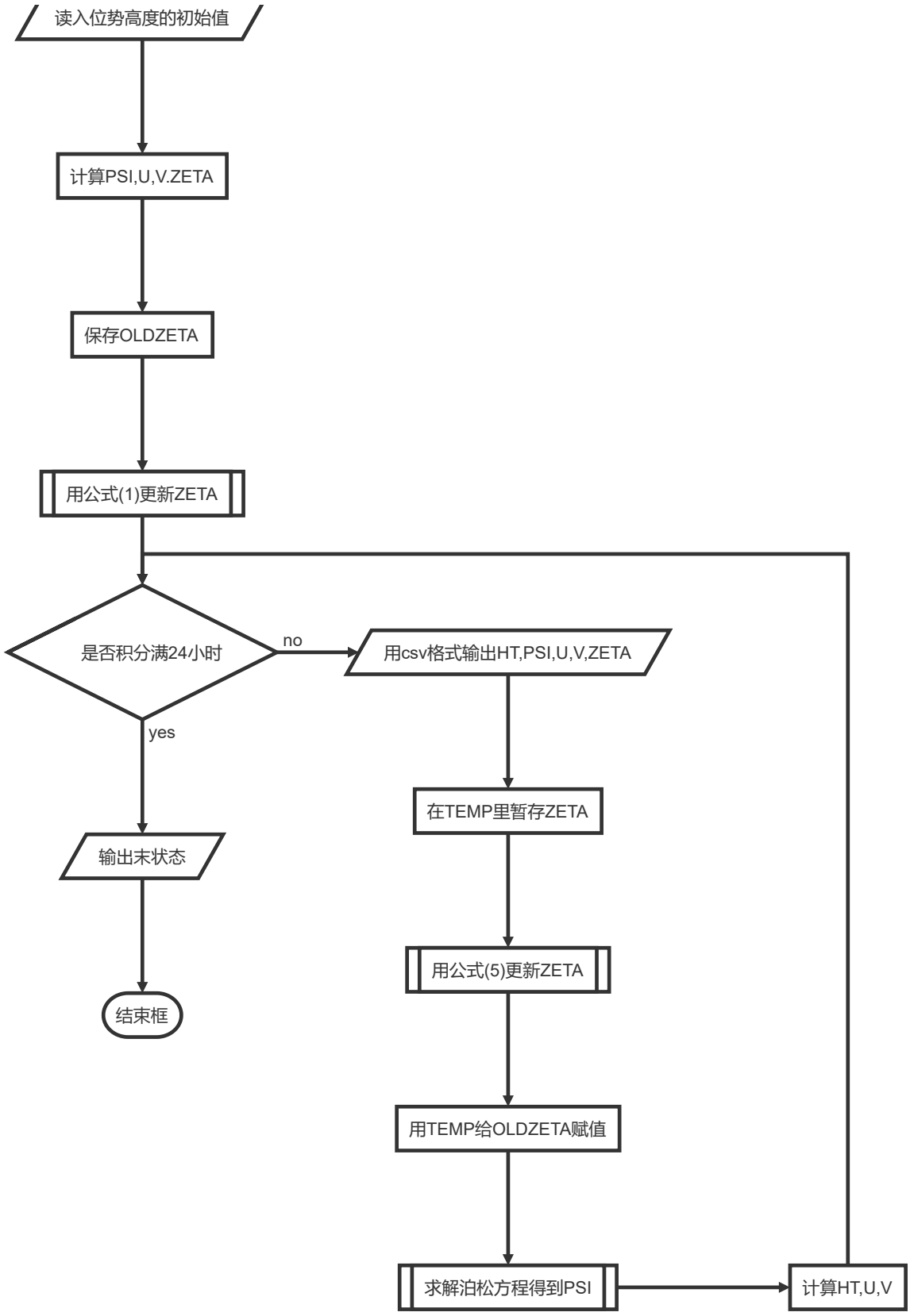
Leap-frog 格式

Leap-frog格式就是把空间差分也写成中心差分的形式，具体表示为

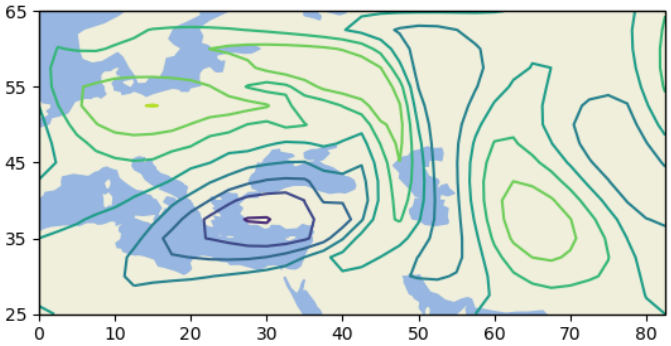
$$\frac{\zeta^{n+1}-\zeta^{n-1}}{2\tau} = -u^n\frac{d_h\zeta^n}{d_hx} - v^n\frac{d_h\zeta^n}{d_hy} + \kappa\nabla_h^2\zeta^n \tag{5}$$

虽然计算量没有增加，但是保存涡度的内存需求大约增加了一倍。流程图也需要做一些调整。





最终预报的涡度场如下图所示



和松野格式不分伯仲。

集合预报

见动画，使用了最为可靠的几个模型。依次为向前差，向前差+平滑（s=0.03），向前差（dt=900），向前差（k=7e4），松野格式，Leap-frog格式。

结论与讨论

此次作业实现了正压涡度方程的向前差求解，预报出了高压向东的移动。在此基础上证明了模式对κ, β, f_0都不是很敏感。为了使计算更加稳定，我们选用了增加时间分辨率、平滑、加大粘性项、改变时间积分格式这几种方法。其中平滑和加大粘性项的尝试并不十分成功

---

1. 张耀科. 正压涡度方程中考虑粘性项与平滑过程的一些意见[J]. 气象学报, 1965 (1): 3. ↩ ↩
2. https://github.com/daleroberts/poisson/blob/master/poisson.py ↩