UNIFIED MODEL DOCUMENTATION NO. ??

Fourier Filtering in the Ocean Model

Unified Model Version 4.5

Graham Rickard and Steve Foreman

The Met. Office, U.K.

Reviewed by : James Penman

March, 2000

The Met. Office,
London Road,
Bracknell,
Berkshire,
RG12 2SY
United Kingdom

Modification History

Date	Author	UM Version	Change Description

Contents

1 INTRODUCTION

Finite difference models on a regular latitude-longitude grid have decreasing zonal gridlength as the poles are approached. This imposes severe constraints on the timestep of an explicit integration scheme if the integration is to remain stable. For example, the Courant-Friedrichs-Lewy (CFL) condition for advection requires,

$$\Delta t < \frac{\Delta x}{c}, \tag{1}$$

where Δt is the timestep, Δx is the gridlength, and "c" the advecting velocity. In order to prevent the stability criteria near the poles dominating the numerical procedures it is usual to impose some form of filtering of the model fields, thus reducing the effective gridlength.

One filtering technique is known as Fourier chopping (Williamson, 1976) in which the fields to be smoothed are Fourier analysed and waves shorter than the minimum stable gridlength discarded. This scheme has been used in numerical weather prediction (NWP) models, but was found to introduce roughnesses in the model fields at a wavelength near to the minimum retained (Bell, 1984). A scheme was introduced which selectively damped (rather than truncated) the unstable wavelengths, so-called Fourier damping. This was found to produce smoother fields (Dickinson, 1985), and is used in the present operational NWP model at the Met. Office (Cullen et al 1993).

The ocean model (Cox, 1984) has always used Fourier chopping, and the aim of this paper is to describe the way in which this has been implemented. A method akin to the Fourier damping of the NWP models is also detailed for completeness, even though it has never been used. In NWP models Fast Fourier Transforms (FFTs), for which many efficient algorithms exist (e.g. Temperton 1983), are used. The efficiency of the FFT algorithms, especially on vector computers, relies on processing arrays of identical length. Latitude-longitude models of the atmosphere use the technique to filter around parallels of longitude, which are the same length at all latitudes and heights. Ocean models, though, filter within ocean basins, so the number of points in the filter varies with geographical position. The Cox (1984) model uses an algebraic transform for this filtering, the details of which are included in this paper.

This documentation represents an updated version of an Ocean Applications internal paper by Foreman (1995) (hereafter referred to as F1) to take account of developments related to the filtering arising from the move to parallel architecture after F1 appeared. Furthermore, the advent of the eddy-permitting ocean model using a resolution of 1/3X1/3 degrees brought the time spent in the filtering routines to prominence. Indeed, the original form of the filtering accounted for up to fifty per cent of the total run time. This contrasts strongly with the present ocean component of HadCM3 which has a resolution of 1.25X1.25 degrees, where the time spent filtering is not significant.

2 Fourier Chopping

Numerical stability of the ocean model is ensured by removing all numerically unstable scales of motion from the model fields. This is done using the method of Fourier chopping. As in F1, the technique is described using the one-dimensional advection equation,

$$\frac{\partial a}{\partial t} = c\frac{\partial a}{\partial x}. \tag{2}$$

In finite difference form (leap-frog, centred differences) this becomes,

$$a_j^{n+1} - a_j^{n-1} = \frac{c\Delta t}{\Delta x}(a_{j+1}^n - a_{j-1}^n). \tag{3}$$

Consider the solution a_j^n to be of the form,

$$a_j^n = \zeta_j^n exp(i(jk\Delta x)). \tag{4}$$

Equation (3) then becomes

$$\zeta^2 - 1 = 2i\zeta r\sin(k\Delta x), \tag{5}$$

where $r = (c\Delta t)/\Delta x$, i.e.,

$$\zeta = ir\sin(k\Delta x) \pm \sqrt{(1 - r^2\sin^2(k\Delta x))}. \tag{6}$$

Thus, if $|r| \leq 1$, $|\zeta| = 1$, and the wave does not grow. More generally the requirement is that

$$|r\sin(k\Delta x)| \leq 1 \tag{7}$$

for a wave to be stable. Fourier chopping imposes this condition by removing all wave components for which this relationship does not hold.

The standard technique used to implement Fourier chopping is (for all latitudes where $|r| > 1$ for some waves) to determine the wave components using a discrete Fourier transform, discard all unstable waves, and re-combine the remaining waves to form the filtered field. This is implemented in the ocean model using a technique which relies on the series involved being summed analytically, as described in Appendix B.

3 Technical Details : Ocean Filtering Subroutines

At the time of writing, the filtering in the ocean code remains of the chopping rather than damping variety. The move to parallel architecture has also meant some re-organisation of the code in order to filter in an efficient ("load-balanced") way. There are some new routines described below to do this. However, at the heart of the filtering rests OFILTR.dk, which as far as can be told is in its original form.

The routine organising the filtering of the tracers and the velocities is OFLTCNTL.dk. Since the filtering operates only in the vicinity of the poles it would clearly be inefficient to do the filtering on the basis of the original ocean row-by-row decomposition; most of the processors would remain idle waiting for those near to the poles to complete the filtering. Thus the work is distributed over all the processors (PEs), and the filtered fields returned to their original PEs afterwards.

How the work is distributed is set up in routine DECMFLTR.dk. Here the rows to be filtered and their relative lengths are assessed in order to ensure that all PEs work equally hard at the filtering. This decomposition is then put into action by OFLTCNTL.dk from a call in TRACER.dk to deal with tracers and velocities. The streamfunction parameter ZTD is also filtered, but via a direct call to OFILTER.dk from BLOKCALC.dk.

The filtering algorithms are given by equations (A3), (A10), and (A13). Each of these depends on N the number of points to be filtered, and p the number of modes to be kept. A table of coefficients thus needs to be set up for the analytic result from equation (B4), one coefficient for each grid point to be filtered. In OFILTR.dk, the table is stored in the array FTARR. The whole formalism is based on combinations of cosines; these combinations are setup at the outset and stored for reference in the arrays COSSAV and DENMSV. For a given (p, N), the numerator and denominator to give FTARR are then built out of COSSAV and DENMSV, and stored in the elements COF and DENOM, respectively.

The value of p depends on the latitude ϕ of the row to be filtered. At a given reference latitude ϕ_0, say, all modes are to be retained. Thereafter,

$$p = N\frac{\cos\phi}{\cos\phi_0}, \tag{8}$$

i.e, as the poles are approached fewer of the modes are kept.

Time is expended in setting up FTARR. Thus OFLTCNTL.dk passes pairs (either u and v or T and S) for a given (p, N), so that on the first pass FTARR is set up, but on the second pass it is not. This is determined by the parameter ISS in OFILTR.dk, with ISS $= 0$ to set up FTARR, and ISS > 0 to not do so.

To unscramble how OFILTR.dk works, the following results are required based on Appendix B, starting with the basic result for the sum of a cosine series, viz,

$$\sum_{k=P}^{Q}\cos k\alpha = -\left(\frac{[\cos(P-1)\alpha - \cos P\alpha] + [\cos(Q+1)\alpha - \cos Q\alpha]}{2(1-\cos\alpha)}\right). \tag{9}$$

Examination of the transforms in Appendix A shows that terms are required for which the coefficient $\alpha = 0$. In this limit the sum in equation (9) simply reduces to,

$$\sum_{k=P}^{Q} 1 = Q - P + 1. \tag{10}$$

Now consider the cosine transform. The form detailed in Appendix A takes the sum from $k = 1$ to $k = p$. However this is inconsistent with the coding in OFILTR.dk.

This is due to the mean term $\bar{f}$ which effectively accounts for the term $k=0$, so that the sum should be over $k=1$ to $k=p-1$. We therefore can write,

$$g(x_m) = \bar{f} + \frac{2}{N}\sum_{j=1}^{N} f(x_j)\left[\left(-\frac{1}{4} - \left[\frac{\cos(p+1)E - \cos pE}{4(1-\cos E)}\right]\right)\right]_{j\neq m}\left[\left(\frac{p-1}{2}\right)\right]_{j=m}$$

$$+\frac{2}{N}\sum_{j=1}^{N} f(x_j)\left(-\frac{1}{4} - \left[\frac{\cos(p+1)F - \cos pF}{4(1-\cos F)}\right]\right). \qquad (11)$$

The factors involving the cosines in the square brackets are stored in FTARR. It is also arranged that the contribution involving $\cos E$ is zero when $j = m$ by setting DENOM to zero for these elements. In its full glory, FTARR is a NXN matrix, so that the elements $j = m$ lie on the matrix diagonal. In the code all elements of FTARR are first set, with the factor FXA $= 1/2$ to account for the two factors of $1/4$ in equation (11). A separate loop then corrects all the diagonal elements by adding back C1 $= (p-1)/2 + 1/4$.

The exercise can be repeated for the sine and full transforms. For the sine transform we can write,

$$f_f(x_k) = \frac{2}{(N+1)}\sum_{i=1}^{N} f(x_i)\left(\frac{1}{4} + \left[\frac{\cos(p+1)F - \cos pF}{4(1-\cos F)}\right]\right)$$

$$+\frac{2}{(N+1)}\sum_{i=1}^{N} f(x_i)\left[\left(-\frac{1}{4} - \left[\frac{\cos(p+1)E - \cos pE}{4(1-\cos E)}\right]\right)\right]_{i\neq k}\left[\left(\frac{p}{2}\right)\right]_{i=k}. \qquad (12)$$

In this case the correction to the diagonal terms is now C1 $= p/2 + 1/4$. The full transform reduces to,

$$f_f(x_m) = \frac{a_0}{2} + \frac{2}{N}\sum_{k=1}^{N} f(x_k)\left[\left(-\frac{1}{2} - \left[\frac{\cos(p+1)E - \cos pE}{2(1-\cos E)}\right]\right)\right]_{k\neq m}[(p)]_{k=m}. \qquad (13)$$

The diagonal correction term is now C2 $= p + 1/2$. If the original signal is to be recovered from the full transform when $p = N/2$ for N even, an extra term is required in equation (13), as detailed by equation (A13).

Having obtained FTARR, all that is left is to perform the sums on the right hand sides of equations (11) to (13), and repeat this for each grid point in the row being filtered. This appears ultimately as a matrix multiply in the code in the form,

$$SPRIME_{(j)} = \sum_{i=1}^{N} FTARR_{(i,j)} S_{(i)}, \tag{14}$$

for j from 1 to N. SPRIME thus represents the filtered signal. There is one more check, however, and that is to ensure that the filtered signal does not have a different mean value from the original signal. Note that the boundary condition of zero at either end of the row for the sine transform implies that the mean in the original signal cannot be altered by the filtering process. The converse is true for the cosine and full transforms. Thus for the latter two, the code initially subtracts off the mean of the original signal (in variable $STEMP = SSUM/N$), obtains SPRIME and its sum SSM, and returns the final filtered signal SFINAL via,

$$SFINAL_{(j)} = \frac{(SSUM - SSM)}{N} + SPRIME_{(j)}. \tag{15}$$

3.1 Filtering a Vector Field in the Ocean Code

Examination of OFLTCNTL.dk will reveal that the filtering of the velocity field (u, v) is actually performed on new variables $UDIF$ and $VDIF$. The filtered (u, v) is obtained from the filtered values of $UDIF$ and $VDIF$. On the surface of a sphere along constant latitude circles, the variation of the components does not quantify how noisy the field is. It is necessary to choose a cartesian reference frame in which to view the field. For example, a field with a constant u (zonal) component and zero v (meridional) component would appear to be entirely smooth in spherical coordinates, but projected onto a two-dimensional plane is actually relatively noisy.

The cartesian components used in the ocean code are shown in Figure 2. The top view is for the Northern hemisphere, the lower for the Southern hemisphere. Each frame shows the relative directions of the velocity field as viewed from above each respective pole. The meridional component v points to the North Pole, hence v pointing away from the South Pole in the lower frame. The angle θ here is that referred to in the ocean code through the variables SPCOS ($\cos\theta$) and SPSIN ($\sin\theta$), with the convention that θ increases in the direction of u. The dashed lines in Figure 2 are local cartesian axes on to which the components can be projected. The particular choices made by the ocean code are explicitly included.

4 Technical Details : Speeding Up the Filtering

As highlighted previously, it was the development of the eddy-permitting version of the ocean model that drew attention to the filtering routines. The typical row length in HadCM3 is 288 grid points with 144 rows, whereas the eddy-permitting model requires 1080 grid points per row and 570 rows. This resolution increase resulted in the filtering becoming almost half the total run time of the code, with the total run time itself being well in excess of that required to perform meaningful climate studies. The filtering burden clearly needed to be reduced significantly.

From the notes above, it is perhaps no surprise that the filtering became noticeably expensive. The array FTARR comprises NXN elements, and therefore scales as N^2, and not only has there been a near four-fold increase in the row length, but the number of rows to be filtered has also increased. On top of the time element, there was also an extra memory burden since (as noted above) the filtering routines carry around FTARR, COSSAV, and DENMSV from timestep to timestep. The memory used by these arrays added quite significantly to the eddy-permitting model's overall requirement.

Examination of FTARR and the formulae used to determine it (see equations (11), (12), and (13)) reveal a simple dependence on combinations of cosine terms. The inherent symmetry of the cosine terms can then be exploited to show that the array FTARR is symmetric about both diagonals. Thus, in principle, it is not necessary to calculate all of the NXN elements as the reference version of OFILTR.dk does, but only to do a quarter of the work, and fill the remaining elements as required.

Delving further in to the workings of OFILTR.dk, it transpires that the matrix multiply in equation (14) takes up most of the time. This is evident from Table 1,

Table 1: Single Timestep Relative Timings

Section	Control	Half	Quarter	Half SSYMV	Half SSPMV
OFILTR	17.8	11.3	8.19	3.74	3.6
(1) COS	1.38	0.635	0.314	0.632	0.583
(1) SIN	1.26	0.552	0.275	0.54	0.501
(1) FULL	2.54	1.06	0.538	1.08	1.03
(1) TOTAL	5.18	2.25	1.13	2.25	2.11
(2) SPRIME	11.1	8.98	5.68	1.34	1.16

which shows relative timings within OFILTR for a single timestep. The first column shows the original (Control) model timings within OFILTR itself, then for each of the separate decompositions within OFILTR to set up FTARR for COSine, SINe, and the FULL transforms followed by their TOTAL. The bottom row SPRIME indicates the time taken to do the matrix multiply. SPRIME here accounts for over two-thirds of OFILTR, with TOTAL accounting for the rest.

Column 'Half' gives the timings obtained by exploiting the symmetry about the main diagonal. Clearly TOTAL is down by a half, and some re-ordering within SPRIME has reduced the time there also. Column 'Quarter' now exploits the symmetry about both diagonals, halving TOTAL again, and having a further impact on SPRIME. Quarter has now halved the Control run time. It might be thought that SPRIME should be going down linearly. However, to perform the matrix multiply requires judicious cycling of arguments to pick out relevant terms from those stored. This introduces an overhead in calculating indices and then fetching the variable from the array. There may well be room for more improvement here.

Optimisation of such an operation will depend, of course, on the machine being used. For the T3E parallel architecture used at the time of writing, there are specific packages designed to deal with linear algebra of the form implied by SPRIME, so-called 'BLAS' routines. One package called SSYMV performs the matrix multiply

Table 2: Five Day Ocean Only Timings

Section	Control	Half	Quarter	Half SSYMV	Half SSPMV
OFILTR TIME	137	76	55	38	36
TOTAL TIME	668	568	550	540	527

using an NXN matrix. There is not a package to deal with the double symmetry of FTARR. However, there is a routine that exploits the symmetry about the leading diagonal, called SSPMV. The 'P' here refers to 'Packed', in that only the upper diagonal elements need be provided.

In the experiments here it was found that filling the matrix elements exploiting the Quarter symmetry gave no gain over the Half symmetry when using the BLAS routines. This appears again due to the time overhead in transferring calculated elements into their respective slots. Thus the BLAS timings are only recorded as 'Half SSYMV' and 'Half SSPMV'. It is evident that the main impact is on the SPRIME operation, with major speedups evident. The upshot is that our best combination Half SSPMV has produced a five-fold decrease in the OFILTR timings.

To get a feel for the timings on a longer timescale, Table 2 shows results from five day ocean only runs, with a forty level ocean with $1.25X1.25$ degrees horizontal spacing. The numbers now represent total CPU time. Again our best combinations lead to a five-fold decrease in the time spent filtering, with some twenty per cent savings in the total run time. At present, these optimisations have greatly reduced the computational dependence of the eddy-permitting model on the filtering.

If the resolution of the ocean model is to be further enhanced, then the relatively poor scalings associated with filtering are likely to put OFILTR.dk back in the spotlight. The present optimisations have been sufficient to keep the filtering in check (just). However, there remain fewer obvious ways to make it go faster. One possibility is to store FTARR for most combinations of (p, N); while this would certainly give nearly a halving in time, the consequent memory requirement would be enormous. It is clear that future models must do away with filtering, not only from the run-time aspect, but also to avoid the presumably none too insignificant impacts of filtering on the physical solution being sought.

Acknowledgements

Thanks to Helene Banks, Doug Cresswell, and James Penman for insightful unscrambling of both the algorithms and algebra. Richard Hill was responsible for the coding of the decomposition for the efficient processing of the filtering on parallel architecture.

References

Bell, R.S. 1984: Filtering in the Met. Off. global forecast model, *Met O 11 working paper No. 70.*

Bryan, K 1984: Accelerating the convergence to equilibrium of ocean-climate models, *J.P.O*, **14**, 666-673.

Cox, M.D. 1984: A primitive equation model of the ocean, *GFDL Ocean Modelling Group Report No. 1*

Cullen, M.J.P., T. Davies, and M.H. Mawson 1993: Conservative finite difference schemes for a unified forecast/climate model, *Unified Model Documentation Paper No. 10*

Dahlquist, G. and A. Bjorck 1984: Numerical Methods (Prentice Hall).

Dickinson, A. (ed) 1985: The weather prediction model, *Operational Weather Prediction System Documentation Paper No. 4.*

Foreman, S.J. 1995: Fourier filtering in the GFDL ocean model, *Ocean Applications Internal Paper No. 6.*

Temperton, C. 1983: Self-sorting mixed-radix Fast Fourier Transforms, *J. Comp. Phys.*, **52**, 1-23.

Williamson, D.L. 1976: Linear stability of finite difference approximations on a uniform latitude-longitude grid with Fourier filtering, *Mon Wea Rev*, **104**, 31-41.

Appendix

A Fourier Transform Formulae for the Ocean Model

A.1 Fourier Cosine Transform

Only values at tracer gridpoints are filtered using a cosine series, and therefore the values are at points $x_m = (m - 1/2)\Delta x$, where $\Delta x = \pi/N$ (there are N points to be filtered).

Figure 1 shows how this arises. The boundary conditions of $\partial T/\partial x = 0$ and $\mathbf{u} = \mathbf{0}$ are imposed at the velocity points adjacent to the first ocean tracer points along a given row (here labelled $U = 1$ and $U = 5$). The transform is taken to be over the interval including the boundary velocity points, thus the tracer points are staggered by half a grid point relative to the full grid velocity points. Clearly, the boundary conditions determine the transform applicable to each variable. In the example shown in Figure 1, there are 4 tracer points to be filtered, hence $N = 4$. If the velocity points are to be filtered, then $N = 3$ instead.

Let $E = (m - j)\Delta x$ and $F = (m + j - 1)\Delta x$. The basic transform is defined by,

$$a_k = \frac{2}{N}\sum_{j=1}^{N} f(x_j)\cos kx_j, \qquad (\text{ A1 })$$

and,

$$g(x_m) = \frac{1}{N}\sum_{j=1}^{N} f(x_j) + \sum_{k=1}^{p} a_k \cos kx_m, \qquad (\text{ A2 })$$

$$\begin{aligned}
&= \bar{f} + \frac{2}{N}\sum_{k=1}^{p}\cos kx_m \sum_{j=1}^{N} f(x_j)\cos kx_j \\
&= \bar{f} + \frac{2}{N}\sum_{j=1}^{N} f(x_j)\sum_{k=1}^{p}\cos kx_m \cos kx_j \\
&= \bar{f} + \frac{2}{N}\sum_{j=1}^{N} f(x_j)\sum_{k=1}^{p}\left(\frac{\cos kF + \cos kE}{2}\right),
\end{aligned}$$

using the identity

$$\mathrm{cos}a\mathrm{cos}b = \frac{1}{2}(\cos(a+b) + \cos(a-b)).$$

The results from Appendix B are now used to give analytic results for the cosine terms summed over the index k. A further simplification is obtained by splitting the sum over the index j in $\mathrm{cos}kE$ as,

$$\frac{2}{N}(\sum_{j=1}^{m-1} + \sum_{j=m+1}^{N} + \sum_{j=m}) \sum_{k=1}^{p} f(x_j)\frac{\mathrm{cos}kE}{2}.$$

When $j = m$, $E = 0$ so that the sum over index k simply gives $pf(x_m)/2$. Putting all this together we can (finally) write,

$$g(x_m) = \bar{f} + \frac{p}{N} f(x_m)$$

$$+\frac{2}{N}[\sum_{j=1}^{m-1} + \sum_{j=m+1}^{N}] f(x_j)\left(-\frac{1}{4} - \frac{\cos(p+1)E - \mathrm{cos}pE}{4(1-\mathrm{cos}E)}\right)$$

$$+\frac{2}{N}\sum_{j=1}^{N} f(x_j)\left(-\frac{1}{4} - \frac{\cos(p+1)F - \mathrm{cos}pF}{4(1-\mathrm{cos}F)}\right), \qquad \text{(A3)}$$

i.e.,

$$g(x_m) = \bar{f} + \frac{p}{N} f(x_m) - \frac{2}{N}[\sum_{j=1}^{m-1} + \sum_{j=m+1}^{N}] f(x_j)\left(\frac{1}{4} + \frac{\cos(p+1)E - \mathrm{cos}pE}{4(1-\mathrm{cos}E)}\right)$$

$$-\frac{2}{N}\sum_{j=1}^{N} f(x_j)\left(\frac{1}{4} + \frac{\cos(p+1)F - \mathrm{cos}pF}{4(1-\mathrm{cos}F)}\right).$$

These formulae can be checked by setting $p = N$, i.e, retain all the Fourier components so that the original signal should be recovered. Using,

$$\cos(N+1)j\Delta x - \mathrm{cos}Nj\Delta x = \mathrm{cos}j\pi(\mathrm{cos}j\Delta x - 1)$$

for integer j gives,

$$g(x_m) = \bar{f} + f(x_m) - \frac{1}{2N}\sum_{j=1}^{N} f(x_j)(1-(-1)^{m-j}) - \frac{1}{2N}\sum_{j=1}^{N} f(x_j)(1-(-1)^{m+j-1}).$$

Rearranging the sums results in,

$$g(x_m) = f(x_m) + \bar{f} - \frac{1}{N}\sum_{j=1}^{N} f(x_j) + \frac{1}{2N}\sum_{j=1}^{N} f(x_j)(-1)^m[(-1)^{-j} - (-1)^j].$$

Recalling the definition of $\bar{f}$ it is then clear that we simply recover the desired result, viz $g(x_m) = f(x_m)$. Thus the basic equations (A1) and (A3) do indeed represent the Fourier cosine transform.

A.2 Fourier Sine Transform

Only values at velocity gridpoints are filtered using a sine series, and the values are at points $x_k = k\Delta x$, $k = 1$ to $k = N$, where $\Delta x = \pi/(N+1)$. Figure 1 shows how the counting arises on the velocity grid points (recalling the boundary conditions $\mathbf{u} = \mathbf{0}$ at land points).

Here define $E = (i-k)\Delta x$, $F = (i+k)\Delta x$.

The discrete sine transform is (e.g. Dahlquist and Bjorck, 1974),

$$b_j = \frac{2}{(N+1)}\sum_{i=1}^{N} f(x_i)\mathrm{sin}ij\Delta x, \qquad (\text{ A4 })$$

$$f(x_k) = \sum_{j=1}^{N} b_j \mathrm{sin}jk\Delta x. \qquad (\text{ A5 })$$

The filtered values are given by,

$$f_f(x_k) = \sum_{j=1}^{p} b_j \mathrm{sin}jk\Delta x. \qquad (\text{ A6 })$$

Thus

$$f_f(x_k) = \frac{2}{(N+1)}\sum_{i=1}^{N} f(x_i)\sum_{j=1}^{p}\mathrm{sin}jk\Delta x\mathrm{sin}ij\Delta x. \qquad (\text{ A7 })$$

Now

$$\mathrm{sin}ij\Delta x\mathrm{sin}jk\Delta x = \frac{1}{2}[\mathrm{cos}Ej - \mathrm{cos}Fj], \qquad (\text{ A8 })$$

so that

$$f_f(x_k) = \frac{1}{(N+1)}\sum_{i=1}^{N} f(x_i)\sum_{j=1}^{p}[\mathrm{cos}Ej - \mathrm{cos}Fj]. \qquad (\text{ A9 })$$

Using the results of Appendix B,

$$f_f(x_k) = \frac{2}{(N+1)}[\sum_{i=1}^{k-1} + \sum_{i=k+1}^{N}]f(x_i)\left(\frac{\cos(p+1)F - \cos pF}{4(1-\cos F)} - \frac{\cos(p+1)E - \cos pE}{4(1-\cos E)}\right)$$

$$+\frac{2}{(N+1)}f(x_k)\left(\frac{p}{2} + \frac{1}{4} + \frac{\cos(p+1)2k\Delta x - \cos p2k\Delta x}{4(1-\cos 2k\Delta x)}\right). \qquad \text{(A10)}$$

This is the form used in the standard model code.

As before, these formulae can be checked by setting $p = N$ in order to recover the original signal. The key here is to use the result,

$$\frac{\cos\left((N+1)M\pi/(N+1)\right) - \cos\left(NM\pi/(N+1)\right)}{\left(1-\cos\left(M\pi/(N+1)\right)\right)} = \cos M\pi,$$

where M is an integer representing either $(i+k)$, $(i-k)$, or $2k$ in equation A10.

A.3 Full Fourier Transform

This applies when there are no land points in the row to be filtered. The row consists of N independent points, such that $\Delta x = 2\pi/N$ and $x_m = (m-1)\Delta x$ with the periodic condition $x_{N+1} = x_1$. For this transform we define $E = (m-k)\Delta x$.

There is an asymmetry in the transform, dependent on whether N is even or odd. From the definitions,

$$f(x_m) = \frac{a_0}{2} + \sum_{j=1}^{\frac{N}{2}-1} a_j \cos j(m-1)\Delta x + \sum_{j=1}^{\frac{N}{2}} b_j \sin j(m-1)\Delta x \qquad N \qquad \text{even},$$

$$f(x_m) = \frac{a_0}{2} + \sum_{j=1}^{\frac{N-1}{2}} (a_j \cos j(m-1)\Delta x + b_j \sin j(m-1)\Delta x) \qquad N \qquad \text{odd},$$

it is clear that when N is even, the cosine sum contains one less term than the sine sum. This has consequences later when attempting to recover the original signal.

The formula for the truncated discrete Fourier transform of these N points is,

$$f_f(x_m) = \frac{a_0}{2} + \sum_{j=1}^{p}(a_j \cos j(m-1)\Delta x + b_j \sin j(m-1)\Delta x), \qquad \text{(A11)}$$

where,

$$a_j = \frac{2}{N}\sum_{k=1}^{N} f(x_k)\mathrm{cos}j(k-1)\Delta x,$$

$$b_j = \frac{2}{N}\sum_{k=1}^{N} f(x_k)\mathrm{sin}j(k-1)\Delta x.$$

Thus,

$$\begin{aligned} f_f(x_m) = \frac{a_0}{2} \quad &+ \quad \frac{2}{N}\sum_{k=1}^{N} f(x_k)\sum_{j=1}^{p}\mathrm{cos}j(m-1)\Delta x\mathrm{cos}j(k-1)\Delta x \\ &+ \quad \frac{2}{N}\sum_{k=1}^{N} f(x_k)\sum_{j=1}^{p}\mathrm{sin}j(m-1)\Delta x\mathrm{sin}j(k-1)\Delta x, \end{aligned}$$

$$\begin{aligned} f_f(x_m) = \frac{a_0}{2} \quad &+ \quad \frac{2}{N}\sum_{k=1}^{N} \frac{f(x_k)}{2}\sum_{j=1}^{p}\mathrm{cos}(m+k-2)j\Delta x + \mathrm{cos}jE \\ &+ \quad \frac{2}{N}\sum_{k=1}^{N} \frac{f(x_k)}{2}\sum_{j=1}^{p}\mathrm{cos}jE - \mathrm{cos}(m+k-2)j\Delta x, \end{aligned}$$

i.e.,

$$f_f(x_m) = \frac{a_0}{2} + \frac{2p}{N}f(x_m) + \frac{2}{N}[\sum_{k=1}^{m-1} + \sum_{k=m+1}^{N}]f(x_k)\sum_{j=1}^{p}\mathrm{cos}jE. \qquad (\text{ A12 })$$

Equation (A12) is exact if the number of cosine and sine terms in the truncated series is the same. If N is odd there are equal numbers of cosine and sine terms (see above), and the original signal can be recovered by setting $p = (N-1)/2$, and using $\sum_{k=1}^{N} f(x_k) = 0$ since the mean is accounted for by a_0. However, if N is even and $p = N/2$ is used, it is clear that an extraneous cosine contribution has been included; in this instance, then, an extra term must be included in equation (A12) in order to ensure recovery of the original signal. The code chooses to subtract,

$$\frac{2}{N}\frac{1}{2}\sum_{k=1}^{N} f(x_k)\mathrm{cos}(k-1)p\Delta x\mathrm{cos}(m-1)p\Delta x = \frac{2}{N}[\frac{1}{2}\sum_{k=1}^{N} f(x_k)(-1)^m(-1)^k]$$

(thus giving $f_f(x_m) = f(x_m)$), i.e.,

$$f_f(x_m) = \frac{a_0}{2} + \frac{2p}{N}f(x_m) - \frac{2}{N}[\sum_{k=1}^{m-1} + \sum_{k=m+1}^{N}]f(x_k)\left(\frac{1}{2} + \frac{\mathrm{cos}(p+1)E - \mathrm{cos}pE}{2(1-\mathrm{cos}E)}\right)$$

$$-\delta_{p,N/2}\frac{2}{N}\sum_{k=1}^{N}\frac{f(x_k)}{2}(-1)^{k+m}. \qquad (\text{ A13 })$$

This extra term is constructed in OFILTR.dk from the variable COSNPI. However, COSNPI is incorrectly coded with,

$$\text{COSNPI(IM)} = \cos(\text{IM }\pi/2),$$

and should simply be,

$$\text{COSNPI(IM)} = \cos(\text{IM }\pi).$$

Again, since $\sum_{k=1}^{N} f(x_k) = 0$, the formulae indeed return the original values.

B Sum of a Cosine Series

OFILTER.dk depends on the following result for the sum S. All transforms have been reduced to such sums over cosines so that this analytic result can be used.

Consider the series,

$$S = \sum_{j=P}^{Q}\gamma^j \cos j\alpha, \qquad (\text{ B1 })$$

for real α and γ ($\alpha \neq 0$). Then,

$$\begin{aligned} S &= \sum_{j=P}^{Q}\gamma^j \Re[\exp ij\alpha] \\ &= \Re\left(\sum_{j=P}^{Q}\gamma^j[\exp ij\alpha]\right), \end{aligned} \qquad (\text{ B2 })$$

so that,

$$S = \Re\left(\frac{\gamma^{Q+1}\exp i(Q+1)\alpha - \gamma^P \exp iP\alpha}{\gamma \exp i\alpha - 1}\right) \qquad (\text{ B3 })$$

using the formula for the sum of a geometric series. Thus,

$$S = \Re\left(\frac{\gamma^{Q+1}[\cos(Q+1)\alpha + i\sin(Q+1)\alpha] - \gamma^P[\cos P\alpha + i\sin P\alpha]}{[\gamma\cos\alpha - 1] + i\gamma\sin\alpha}\right),$$

so that,

$$S = \frac{\gamma^{Q+1}[\cos(Q+1)\alpha(\gamma\cos\alpha - 1) + \sin(Q+1)\alpha\gamma\sin\alpha]}{(\gamma\cos\alpha - 1)^2 + \gamma^2\sin^2\alpha}$$

$$-\frac{\gamma^{P}[\cos P\alpha(\gamma\cos\alpha - 1) + \sin P\alpha\gamma\sin\alpha]}{(\gamma\cos\alpha - 1)^2 + \gamma^2\sin^2\alpha},$$

$$S = \frac{\gamma^{Q+1}[\cos Q\alpha - \cos(Q+1)\alpha] - \gamma^{P}[\gamma\cos(P-1)\alpha - \cos P\alpha]}{1 + \gamma^2 - 2\gamma\cos\alpha},$$

giving the final result in the form,

$$S = -\left(\frac{\gamma^{Q+1}[\cos(Q+1)\alpha - \cos Q\alpha] + \gamma^{P}[\gamma\cos(P-1)\alpha - \cos P\alpha]}{1 + \gamma^2 - 2\gamma\cos\alpha}\right). \qquad \text{(B4)}$$

For most purposes of interest $\gamma = 1$, the lower limit $P = 1$ and the upper limit $Q = p$, where p represents the number of modes to be kept. For these limits the sum becomes,

$$S = -\frac{1}{2} - \left(\frac{[\cos(p+1)\alpha - \cos p\alpha]}{2(1 - \cos\alpha)}\right). \qquad \text{(B5)}$$

C Fourier Damping

The structure of the ocean model places a strong constraint on the computational efficiency of a Fast Fourier Transform, making it uneconomic to implement Fourier damping in its full form. This section describes a compromise technique which will maintain stability with less severe effects than Fourier chopping, and which may be implemented in the ocean model with only a small increase in the calculations at each timestep. It represents, however, a more stringent constraint than the full implementation of Fourier damping.

The filtered field is now taken to be,

$$f_f(k\Delta x) = \sum_{j=1}^{p} b_j \sin(jk\Delta x) + \sum_{j=p+1}^{N} \alpha\gamma^j b_j \sin(jk\Delta x), \qquad \text{(C1)}$$

where α and γ are to be determined. It is readily seen that this is an extension of the chopping technique. The shortest wave fully retained is still given by equation (7), shorter waves being damped, since $\alpha\gamma^j < 1$ is imposed.

In order to determine the values of α and γ, the growth rates of the unstable waves must be determined. This is done by considering the numerical stability problem as in Annex C of F1. Let the growth rate of a wave of the form $\sin(jk\Delta x)$ be A_k. Then, for stability to be assured,

$$\alpha\gamma^k \leq \frac{1}{A_k} \qquad \text{for} \qquad p < k \leq N. \tag{C2}$$

Clearly, with only the values α and γ available to fit these constraints some form of optimal values must be chosen. One method is suggested here. Define,

$$d_k = \frac{1}{A_k} \qquad \text{and} \qquad D = \{d_k : p < k \leq N\}, \tag{C3}$$

$$\alpha = max(D). \tag{C4}$$

Then set,

$$\gamma = min(1/(\alpha A_k)^{1/k}) \qquad : \qquad p < k \leq N. \tag{C5}$$

This choice of α and γ will preserve the property $\alpha\gamma^k \leq 1/A_k$ (except for truncation errors) while maintaining a larger value of γ than if $\alpha = 1$. Note that in most cases $\alpha = 1/A_{p+1}$ since waves become more unstable as their wavelength decreases.

While equation (C1) will maintain stability there will still be a marked reduction in the amplitude of the first damped waves. This may induce spurious computational modes, although these should be less important than in the case of Fourier chopping due to the retention of shorter waves, albeit at reduced amplitude, and to the absence of the abrupt changes in truncation with latitude. In practice it would be prudent to use

$$(1-\epsilon)\alpha\gamma^k \tag{C6}$$

as the damping factor for $\sin(k\Delta x)$ for some small ϵ, which may be achieved by redefining α, in order to allow for effects not considered in the linear stability analysis.

The appropriate formulae to use may be derived from those of Appendix A, and are given below for completeness.

Cosine Transform

Here, $x_k = (k-1/2)\Delta x$, $E = (k-j)\Delta x$, and $F = (k+j-1)\Delta x$.

$$f_f(x_k) = \bar{f} + \frac{p}{N} f(x_k) -$$

$$\frac{2}{N}[\sum_{j=1}^{k-1} + \sum_{j=k+1}^{N}] f(x_j)[\frac{1}{4} + \frac{\cos(p+1)E - \cos pE}{4(1-\cos E)}]$$

$$-\frac{2}{N}\sum_{j=1}^{N} f(x_j)[\frac{1}{4} + \frac{\cos(p+1)F - \cos pF}{4(1-\cos F)}]$$

$$+\left(\frac{\gamma^{N+1} - \gamma^{p+1}}{N(\gamma-1)}\right)\alpha f(x_k)$$

$$+\frac{2\alpha}{N}[\sum_{j=1}^{k-1} + \sum_{j=k+1}^{N}][$$

$$f(x_j)\left(\frac{\gamma^{p+1}(\gamma\cos pE - \cos(p+1)E) + \gamma^{N+1}(\cos(N+1)E - \cos NE)}{4(1-2\gamma\cos E + \gamma^2)}\right)]$$

$$+\frac{2\alpha}{N}\sum_{j=1}^{N} f(x_j)\left(\frac{\gamma^{p+1}(\gamma\cos pF - \cos(p+1)F) + \gamma^{N+1}(\cos(N+1)F - \cos NF)}{4(1-2\gamma\cos F + \gamma^2)}\right) \tag{C7}$$

Sine Transform

Here, $x_k = k\Delta x$, $E = (j-k)\Delta x$, and $F = (j+k)\Delta x$.

$$f_f(x_k) = \frac{2}{N+1}[\sum_{j=1}^{k-1} + \sum_{j=k+1}^{N}] f(x_j)\left(\frac{\cos(p+1)F - \cos pF}{4(1-\cos F)} - \frac{\cos(p+1)E - \cos pE}{4(1-\cos E)}\right)$$

$$+\frac{2}{N+1} f(x_k)\left(\frac{1}{4} + \frac{p}{2} + \frac{\cos 2(p+1)x_k - \cos 2px_k}{4(1-\cos 2x_k)}\right)$$

$$+\frac{2\alpha}{N+1}[\sum_{j=1}^{k-1} + \sum_{j=k+1}^{N}]\left(\frac{\gamma^{p+1}(\gamma\cos pF - \cos(p+1)F) + \gamma^{N+1}(\cos(N+1)F - \cos NF)}{4(1+\gamma^2 - 2\gamma\cos F)}\right)$$

$$-\frac{2\alpha}{N+1}[\sum_{j=1}^{k-1} + \sum_{j=k+1}^{N}]\left(\frac{\gamma^{p+1}(\gamma\cos pE - \cos(p+1)E) + \gamma^{N+1}(\cos(N+1)E - \cos NE)}{4(1+\gamma^2 - 2\gamma\cos E)}\right)$$

$$+\left(\frac{\gamma^{N+1} - \gamma^{p+1}}{\gamma - 1}\right)\frac{f(x_k)}{2} + \frac{\gamma^{p+1}(\gamma\cos 2px_k - \cos 2(p+1)x_k) + \gamma^{N+1}(1 - \cos 2x_k)}{4(1+\gamma^2 - 2\gamma\cos 2x_k)} \tag{C8}$$

Full Series

Here, $x_k = (k-1)\Delta x$, $E = (m-k)\Delta x$, and $\bar{f}$ the mean signal is taken to be zero.

$$f_f(x_m) = -\frac{2}{N}[\sum_{k=1}^{m-1} + \sum_{k=m+1}^{N}]f(x_k)[\frac{1}{2} + \frac{\cos(p+1)E - \cos pE}{2(1-\cos E)}] + \frac{2p}{N}f(x_m)$$

$$-\frac{2\alpha}{N}[\sum_{k=1}^{m-1} + \sum_{k=m+1}^{N}]f(x_k)[\frac{\gamma^{p+1}(\gamma\cos pE - \cos(p+1)E)}{(1+\gamma^2-2\gamma\cos E)}] + \frac{2\alpha}{N}f(x_m)\left(\frac{\gamma^{Q+1}-\gamma^{p+1}}{(\gamma-1)}\right)$$

$$-\frac{2\alpha}{N}[\sum_{k=1}^{m-1} + \sum_{k=m+1}^{N}]f(x_k)[\frac{\gamma^{Q+1}(\gamma\cos(Q+1)E - \cos QE)}{(1+\gamma^2-2\gamma\cos E)}]$$

$$-\frac{2\delta_{Q,N/2}}{N}\alpha\gamma^Q\sum_{k=1}^{N}f(x_k)(-1)^{m+k}, \qquad \text{(C9)}$$

where $Q = N/2$ or $Q = (N-1)/2$ for N even or odd, respectively, and $\delta_{a,b}$ is zero or one if $a \neq b$ or $a = b$, respectively.

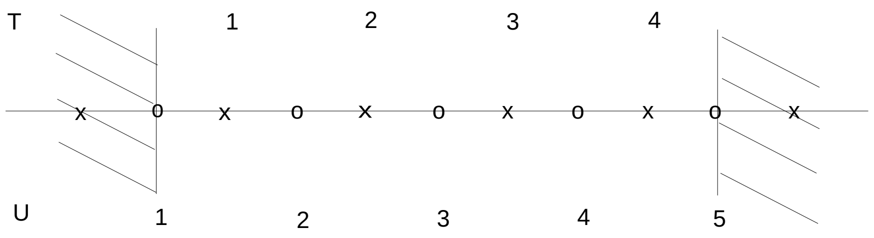

Figure 1: Grid point definitions for a filtered row. The tracer points (T) are labelled by x and the velocity points (U) by o. The hashed areas locate land. The numbers above and below label the tracer and velocity points on the section to be filtered, respectively.

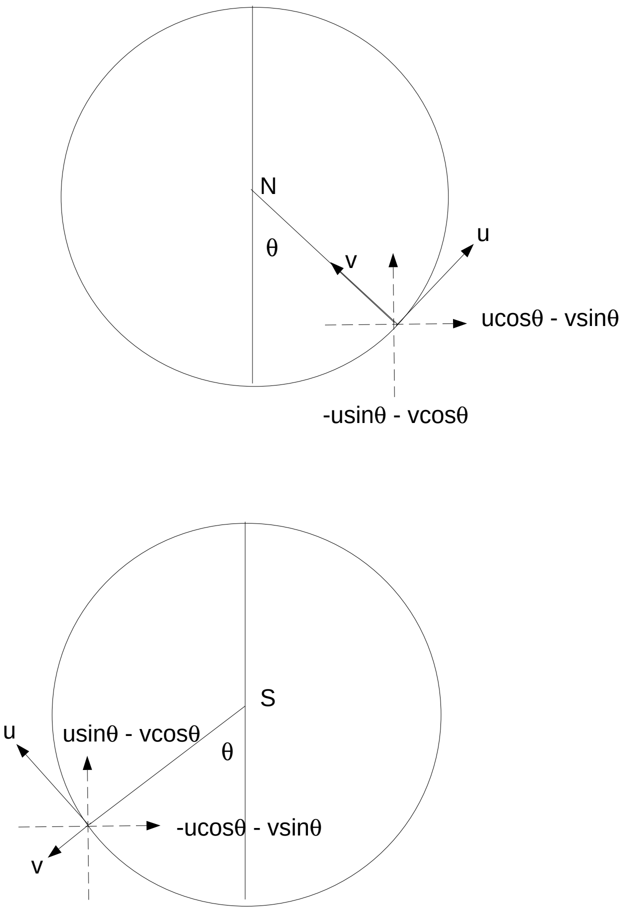

Figure 2: Cartesian components of velocity field for filtering along constant latitude circles.