The Hybrid Coupled Model, Version 3: Technical Notes

David W. Pierce
Climate Research Division
Scripps Institution of Oceanography
August, 1996

Contents

1. Introduction

This note describes the technical details of constructing the statistical atmospheric portion of a hybrid coupled model (HCM), including a model output statistics (MOS) corrector. An HCM is a coupled model consisting of a *full-physics* Ocean General Circulation Model (OGCM) and a *statistical* atmospheric model. The advantage of such a hybrid is that the statistical atmospheric model runs much faster than a full Atmospheric General Circulation Model (AGCM), and (in some places, anyway) can capture much of the variability of the observed data. So an HCM can give useful results without the full expense and complexity of a coupled OGCM/AGCM. Following common usage, in the rest of this note an "HCM" will refer *only* to the statistical atmospheric component of a full hybrid coupled model. A MOS corrector is a way of correcting for mistakes in an HCM's output, and is quite similar in philosophy and construction to an HCM.

This version of the HCM, version 3, is constructed by limiting the Hamburg Ocean Primitive Equation (HOPE) OGCM, version 2, to the north Pacific domain (extending roughly to 30°S). A sponge layer between 30°S and 44°S is used to relax temperature and salinity to observed conditions, as well as to viscously damp out any motion in that region, with relaxation time decreasing and viscosity increasing to the south. The physics of HOPE are unchanged from the stock distribution except that the surface forcing technique has been modified so that the forcing fields take effect on the 15th of the month rather than the first. The standard HOPE configuration applies a month's climatological values on the first on the month, which is a peculiar way of doing things. It also interpolates between forcing values linearly; I substituted cubic spline interpolation.

It should be emphasized that virtually all the information in this note came originally from Tim Barnett, Jack Ritchie, or Tony Tubbs; setting it down here merely compiles it in one place for future reference.

2. Theoretical Basis

2.1. Overview

The theoretical basis for an HCM is straightforward. Start with a description of our objective: what we *have* is a full-physics OGCM; what we *want* are the atmospheric fields to drive that OGCM, such as wind stress, precipitation, and heat flux. Our objective is to predict reasonable atmospheric forcing fields given the instantaneous state of the OGCM.

One approach to doing this is to couple the OGCM to an AGCM which would calculate all such quantities given the ocean model's sea surface temperature (SST),

but the result would be a gigantic (and slow) model. Instead, note that we don't care about many things which the AGCM would predict, such as the detailed three-dimensional structure of the wind and temperature fields. So make the assumption (which will be quantitatively tested later) that the near-surface atmospheric forcing fields are largely *determined* by SST. If true, then you might be able to statistically predict the atmospheric forcing fields based on the model's instantaneous SST field.

The way this prediction works is as follows. Imagine decomposing observed (not model) monthly SST anomalies into a particular set of linearly independent modes, the empirical orthogonal functions (EOFs). EOFs have the property that the first (or strongest) one captures the most variability in the observed field, the second one captures the most of the remaining variability, etc.

You want to keep the "proper" number of modes in this decomposition; the proper number will probably be between 2 and 10 or so. Selecting this number is discussed later. You also want to keep the proper number of time intervals; which is to say, you could form one set of EOFs based on annually averaged data, or four sets of EOFs based on seasonally averaged data, or twelve sets of EOFs based on monthly averaged data. In version 3 of the HCM, everything is kept by month, so there are 12 sets of EOFs. If you end up keeping n modes, then the result is $12n$ total EOFs: n modes for each of the 12 months. Each EOF covers the entire spatial domain of the HCM model, which may be less than the entire domain of the OGCM. In HCM version 3, the HCM is limited to the tropical strip while the OGCM covers the entire north Pacific.

Do a similar decomposition from observations for each of the atmospheric forcing fields which are to be predicted—the zonal wind stress, τ^x, will be used as an example in the remainder of this note. You are then left with $12m$ EOFs for each observed field, where m was the number of modes kept. Note that different observed fields can keep different numbers of modes.

Finally, calculate (by month) the covariance (or correlation—see below) between each of the predictor EOFs and each of the predictand EOFs. These covariances express the observed likelihood that each *pattern* of anomalous τ^x will be associated with each pattern of anomalous SST. This completes the specification of the HCM.

To make a prediction of τ^x using the HCM, you need the following data: the EOFs of the predictor field, in this case the SST anomalies; the EOFs of the predictand field, in this case the τ^x anomalies; the correlations between the predictor and predictand EOFs; and the value of the predictor field, in this case the instantaneous SST anomaly field taken from the OGCM. This last isn't as easy to calculate as it sounds, because really you want it to be the anomaly with respect to the entire

model run. But as you are calculating along, say halfway through the run, how can you know what the SST will be in the future in order to calculate this month's deviation from the entire run's mean? All you can really do is run a long spinup of the model forced by conditions *as similar as possible* to the actual model run, and then calculate the long-time mean fields from that spinup run. Then calculate the model's SST monthly anomaly field by subtracting the model's instantaneous SST field from that month's long term average field taken from the spinup run. Obviously this is a problem if the mean state of the actual run wanders away from the spinup run until the two have different mean surface temperatures. If that happens, you are stuck—you have to try another spinup, or simply realize that applying the HCM in such circumstances will not give good results.

Finally, decompose the model's instantaneous SST anomaly into the amount you have of each SST EOF; then, using the previously calculated covariances, translate these into the amount you have of each τ^x EOF. Sum up these τ^x EOFs, weighted by the amount you have of each one, and that is the anomalous τ^x atmospheric forcing field predicted for the month.

2.2. More Formally

As described in Barnett et al. (1993) , construct the HCM by decomposing the *observed* SST and τ^x monthly anomaly fields, $T_{\rm obs}$and $\tau^x_{\rm obs}$, as follows:

$$T_{\rm obs}(x,y,t) = \sum_n \alpha(t,n)e(x,y,n) \quad (1)$$

$$\tau^x_{\rm obs}(x,y,t) = \sum_m \beta(t,m)f(x,y,m), \quad (2)$$

where $e(x,y,n)$ is the EOF for $T_{\rm obs}$, and is a function of position and mode number n; $f(x,y,m)$ is the EOF for $\tau^x_{\rm obs}$, a function of position and mode number m; α is the "principal component" (PC) of SST, and is a function of time t and mode n; and β is the PC of $\tau^x_{\rm obs}$.

Define the regression coefficient between PC mode n of $T_{\rm obs}$ and PC mode m of $\tau^x_{\rm obs}$ as

$$C(n,m) = \frac{\langle \alpha(t,n)\beta(t,m)\rangle}{\langle \alpha(t,n)^2\rangle}, \quad (3)$$

where the angle brackets denote an average over all t.

At this point the HCM is completely specified—it consists of $e(x,y,n)$, $f(x,y,m)$, and $C(n,m)$.

To make a prediction using the HCM, first form the model's instantaneous SST anomaly, T_{mod}. Then calculate the predictor SST PCs:

$$\hat{\alpha}(t,n) = \sum_{x,y} T_{\mathrm{mod}}(x,y,t) e(x,y,n), \tag{4}$$

and how well these project onto the predicted field's PCs:

$$\hat{\beta}(t,m) = \sum_{n} C(n,m) \hat{\alpha}(t,n). \tag{5}$$

Finally, the predicted wind field is:

$$\hat{\tau}(x,y,t) = \sum_{m} \hat{\beta}(t,m) f(x,y,m). \tag{6}$$

3. Important Procedural Points

This section gives important procedural points which must be understood when constructing the HCM.

3.1. The HOPE Grid

Were someone to devise the most difficult-to-use, error-prone grid possible for an OGCM, they would probably come up with the HOPE grid. Not only is the grid in a staggered diamond shape (technically, an Arakawa E-grid) but it is further split off into two disjoint arrays, the "even" grid and the "odd" grid. This means that all the model code has to be written twice, once for each of the two grids. And on top of all that, the grids are flipped in the North-South direction, so that everything in the model starts in the north.

This all makes anything to do with the grids highly error-prone, and the grids are a fruitful place to search for problems if the HCM is not performing as expected. One thing to watch out for in particular is the fact that the even scalar points, even vector points, odd scalar points, and odd vector points are *all on different grids*. This is illustrated in Figure 1. Each of these four grids has its own set of latitudes and longitudes, and its own land/sea mask, which is different from the other three grids. Another thing to watch out for is that most observed climatology files have to be flipped in the north-south direction before they can be applied to the HOPE model...however, if you are forming climatology from a HOPE run, then either it can't be flipped, or it must be flipped twice. When debugging, the model even and odd data fields should *always* be viewed together. It is easy for one field to be fine and the other completely wrong.

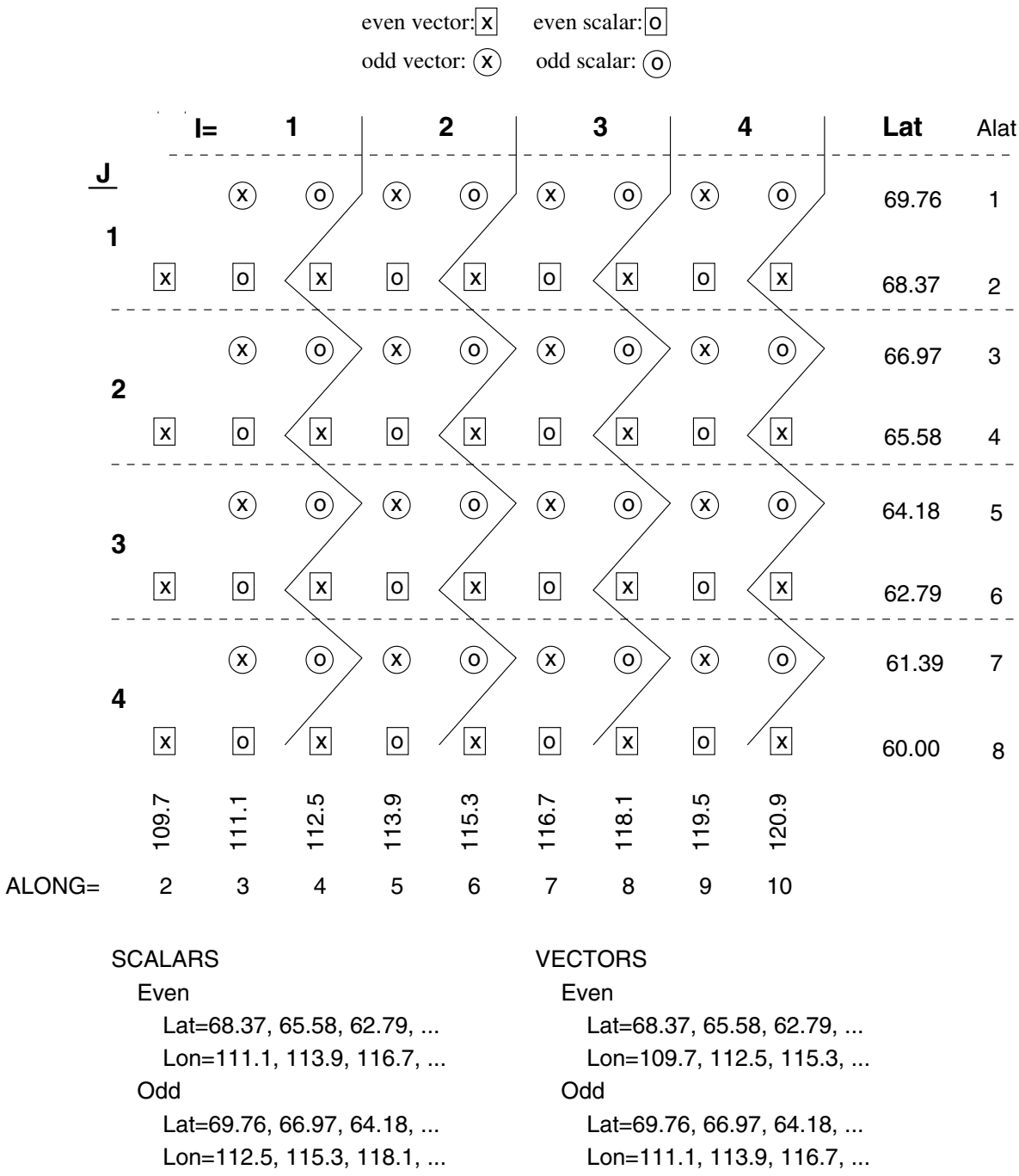

Figure 1: HOPE/HCM version 3 grid.

Never underestimate the amount of damage having quantities on the wrong grid can do. You might think that since the even and odd data values should be pretty close, mixing them up would cause only minor problems; this simply isn't true. It can cause quite major and bizarre performance problems with the HCM and the MOS.

Always apply the correct land/sea mask the input and output of code you add. Frequently, variables over land will have bizarre values which will completely throw off any calculations which are contaminated by them. You *must* mask the output of your routines because later code might not mask their inputs. For safety, you should generally mask your input too, unless you know for certain that it has already been masked.

3.2. Selecting the Data Set

The HCM is based on observed correlations, so you will need observed data sets for the field you are using to predict *from* (the predictor data set) and the data set you are trying to predict *to* (the predictand data set).

For sea surface temperature (SST), most data ultimately devolves out of the Comprehensive Ocean-Atmosphere Data Set (COADS). da Silva's data set is apparently an objectively analyzed, globally gridded version of COADS which goes back to 1945. Reynold's data set (also called the *CAC* (for Climate Analysis Center) or *blended* data set) is another possible source, as is the NMC data set. I do not know the details of the differences between these data sets. I personally use the da Silva version of COADS because it is the newest (and so presumably avoids any past mistakes which could have occurred) and is already gridded.

For wind stress, there is again a da Silva version of COADS winds as well as an independent set, the Florida State University (FSU) winds. The da Silva version has global coverage back to 1945, while the FSU winds have only tropical coverage back to 1977. (Of course, the number of observations must be kept in mind when considering the global coverage back to 1945!) The choice here is trickier, and gets into the next topic (smoothing the data). For instance, Fig. 2 shows the correlation between da Silva and FSU τ^x fields. Note that the correlation is far from perfect; particularly worrisome is the disagreement in the eastern tropical Pacific. The disagreement is smaller when the data are smoothed in both space and time before being compared.

Here is what Arlando da Silva had to say about the discrepancy between his data set and the FSU data set:

```
From dasilva@dao.gsfc.nasa.gov Fri Feb 23 14:50:04 1996
Date: Fri, 9 Feb 1996 17:25:33 -0500 (EST)
```

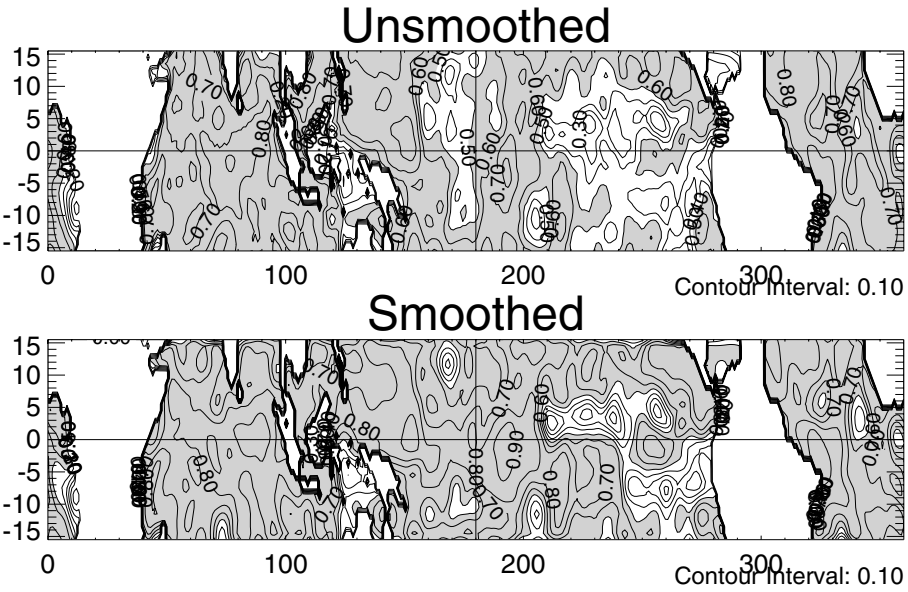

Figure 2: Correlation between the FSU and da Silva datasets for τ^x. Top: unsmoothed. Bottom: smoothed both in time and space with a diffusive algorithm.

From: Arlindo da Silva <dasilva@dao.gsfc.nasa.gov>
To: Dave Pierce <pierce@cirrus.ucsd.edu>
Cc: Arlindo da Silva <dasilva@hera.gsfc.nasa.gov>
Subject: Re: Correlation between FSU & da Silva wind anomalies?

Dear David,

> I want to thank you for making your gridded global data sets available.
> They have made the task of numerical ocean modeling much easier!

You are very welcome!

> I was hoping that I could ask you a question about your wind data sets. In
> the past I've used the FSU winds, which as I'm sure you know don't have the
> same temporal or geographical coverage as the sets you have made and so
> aren't as useful. However, they were available earlier so I started with
> those. I've been working on switching over to using the da Silva winds and
> thought I would see how the two data sets compare to each other. I was
> surprised to find that a correlation of the two data sets (the Tau-X
> anomalies, in particular) shows good agreement in the Atlantic, Indian, and
> western Pacific but not so good agreement in the eastern tropical Pacific,
> with correlations less than 0.5 over most of that region.

I am aware of that. If you look at fig. 4 in

ftp://niteroi.gsfc.nasa.gov/pub/uwm_coads/doc/etc/kiel94_2.ps

```
you will find a similar result.

> I'm sort of
> confounded by this and was wondering if you had some insights into why this
> might be so.  I understand that you apply a stability-dependent transfer
> coefficient to convert the raw wind speed into a drag, while FSU is just
> pseudo-stress, but is that enough to explain all the difference?

I don't think so. The effect of stability correction is usually more
pronounced along western boundary currents. Hellerman & Rosenstein has
a plot showing stress with & without stability correction. Also, since
interannual variability effects on the stability correction are
usually small, I would expect some effect on the climatology, but not
much on anomaly correlation.

> If so, why
> would there still be good correlations in most of the oceans but poor ones in
> the eastern Pacific?
>

Poor data coverage is my best bet. Correlations are usually high along
ship lanes, small in data voids. You find the same in the Atlantic
(see fig. 3 in that Kiel abstract).  When there is no data, we both
"invent something" and since we have different method of analysis
(visual vs. Barnes), the results don't correlate.

> I've made a correlation map for that two data sets that I'm hoping you
> can take a look at if you have a moment.  You can access it via the WWW
> at:
>
> http://meteora.ucsd.edu/~pierce/misc/fsu_daSilva.html
>
> or a Postscript version is available via anonymous ftp from:
>
> cirrus.ucsd.edu:/outgoing/corr_fsu_dasilva_taux.ps.Z
>

Your general patterns agree with ours.

> Thanks for any insights you can share into this issue.

Also our stress anomalies has more noise than FSU's. Jim Carton's
tells me that his ocean model can digest this noise quite nicely, and
he gets realistic interannual variability in both Atlantic and Pacific.
Let me know if you find out that your ocean model respond very
differently to the 2 data sets.

  Best Regards,

     Arlindo.
```

So there you have it. For the results shown here, da Silva's data sets have been used exclusively.

3.3. Selecting the Region

There are subtleties involved in selecting the geographical region to use when constructing the HCM. For HCM version 3, the HCM was constructed from data in a latitudinal band +/- 15.5° around the equator. The entire globe was taken in longitude, despite the fact that HCM version 3 only covers the north Pacific. The reasoning behind this seems to be that the statistical relationship between winds and SST extends throughout the tropical strip and is not confined to the north Pacific. However, in this particular model we were only interested in simulating the north Pacific, and so limited the active model domain to that region.

Another set of subtleties involves the region over which to *apply* the HCM-calculated winds. Obviously this can never be larger than the region used to construct the HCM in the first place; what is not so obvious is that it is advantageous to restrict it to a smaller latitudinal band than the HCM was developed with. In my initial runs, the final HCM-output wind anomalies were restricted to a band +/- 7.5° around the equator. Previous experience has shown that if the region over which the winds are applied is too wide, then the coupled model will tend to oscillate with a strong biennial signal. Later I changed to a mask derived from the skill map of the HCM itself: the winds were only applied in places where the HCM had a skill score ¿ 0.2. Before making this mask I diffused the edges of the skill map rather extensively to avoid introducing anomalous wind curl into the forcing fields. The mask is shown in Figure 3. In the course of debugging, I tried a number of variations on this mask: a narrow version, a wide version, one cut off in the East, one cut off in the West, and one shifted North. The narrow mask had severely diminished SST anomalies; the wider mask had occasional very large cold events; the mask cut off in the East had weaker anomalies but was otherwise similar to the standard; the mask cut off in the West had substantially smaller anomalies; and the mask shifted to the north had only tiny anomalies after the first year.

3.4. Smoothing

Figure 2 shows that smoothing the original data can have a beneficial effect, at least as far as making "observed" data sets agree with each other! It is reasonable to smooth the observed data in both space and time before constructing the EOFs needed for the statistical model so that high frequency noise, which isn't predictable anyway, will not interfere with the EOF calculation.

It is important, however, to smooth the data in the correct way. In particular, boxcar smoothing can be very bad, for two reasons: 1) It leaves the endpoints unchanged, so they must be taken care of manually; 2) It can reverse the phase of periodic signals. These tendencies are illustrated in the top panel of Fig. 4. Note

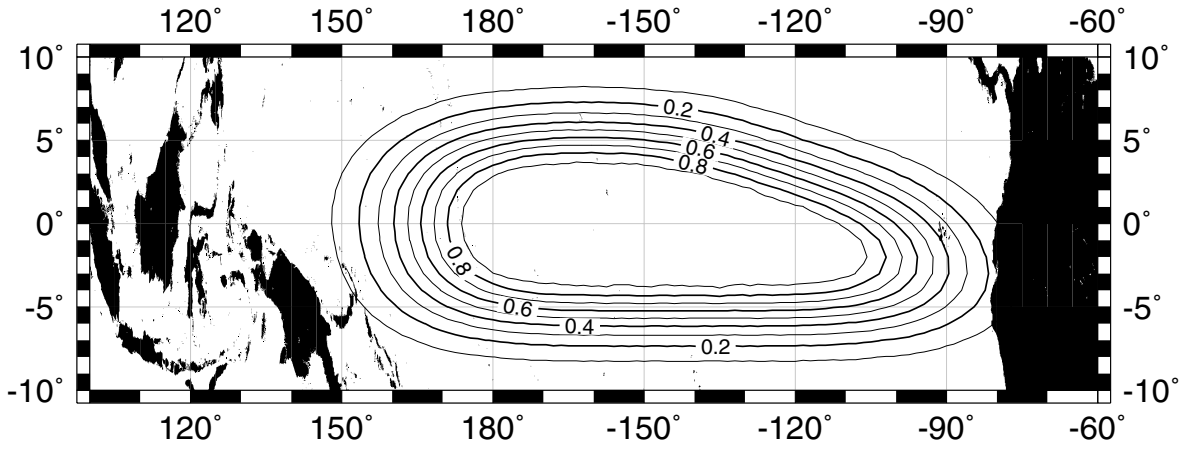

Figure 3: The mask ("tropicmask_v1.2smo") which was applied to the HCM's output winds before those winds forced the ocean model.

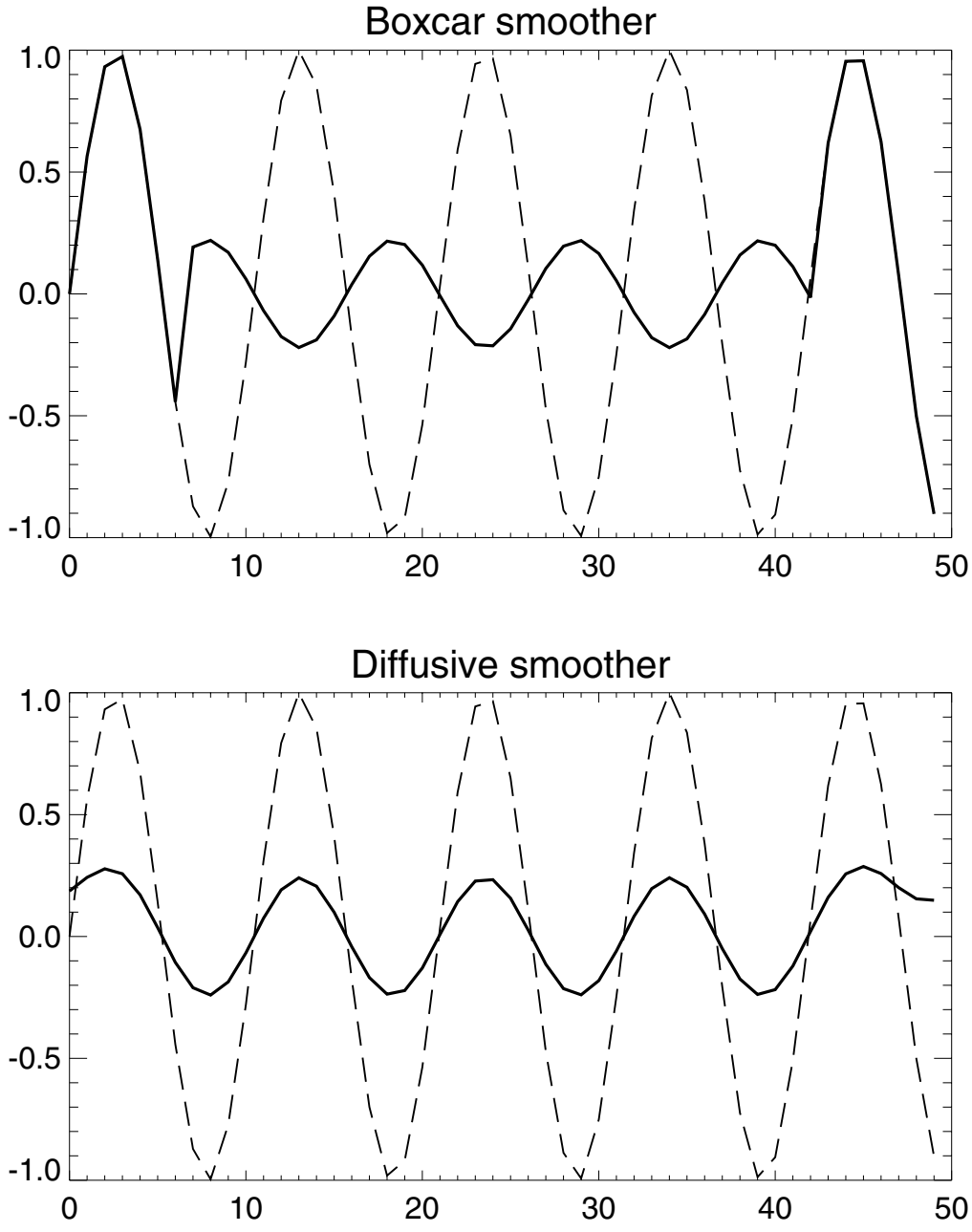

Figure 4: Top: original (dashed) and smoothed (solid) data, smoothed with a boxcar smoother of width 15. Bottom: a diffusive smoother.

that the boxcar-smoothed data is 180 degrees out of phase with the original data! The lower panel shows the data smoothed with a diffusive smoother. There are neither end effects nor the problematic phase inversion which the boxcar smoother shows.

These considerations are not merely academic; Fig. 5 shows τ^x in the middle of the Pacific from the FSU data set. Again, the upper panel shows the boxcar-smoothed data while the lower panel shows data smoothed diffusively. In addition to the end effects, it can be seen that in many places (such as between X=100 and 110) the boxcar-smoothed data is 180 degrees out of phase with the original data set.

The degree to which the data should be smoothed in time is open to some debate. When trying to generate monthly EOFs with monthly observed data, it clearly cannot be too great. For the results shown here, the diffusive smoother was used with the amount of smoothing set to give about the same overall reduction in peak amplitudes as did boxcar smoothing with a total width of 5 months.

Spatial smoothing of the data was also performed, using the same diffusive smoother applied in two dimensions. Again, the amount of smoothing to use is a judgment call; it is desirable to remove some of the spatial noise without significantly modifying the signal.

3.5. Constructing Anomalies

All the EOF routines require anomalies as input, *not* the original data. Constructing monthly anomalies is straightforward: average all the Januaries to form a climatological January, then subtract the climatological January from each individual January. Repeat with the other months, February through December. Note that this is *not* the same as forming the average at each point over all times, then subtracting that average from each individual month.

Most of the data sets, including the FSU and da Silva sets, have already been decomposed into monthly climatologies and anomalies. However — here is the kicker — *you can't use these*. The reason why is actually fairly subtle. Look at Figure 6, which shows SST anomalies over the period 1965 to 1994 from the da Silva data set. Notice anything interesting about the anomalies? In the western regions, especially, the average temperature is clearly varying by more than half a degree on decadal time scales. For example, note how much colder (on average) the temperature between the dateline and 140 W is during 1970 to 1980 than during 1982 to 1992. The da Silva data set was apparently generated by calculating the climatology from the entire period covered, which is 1945 to 1993. However, if you were to run the HCM over a shorter time span, say the 1980s only, then

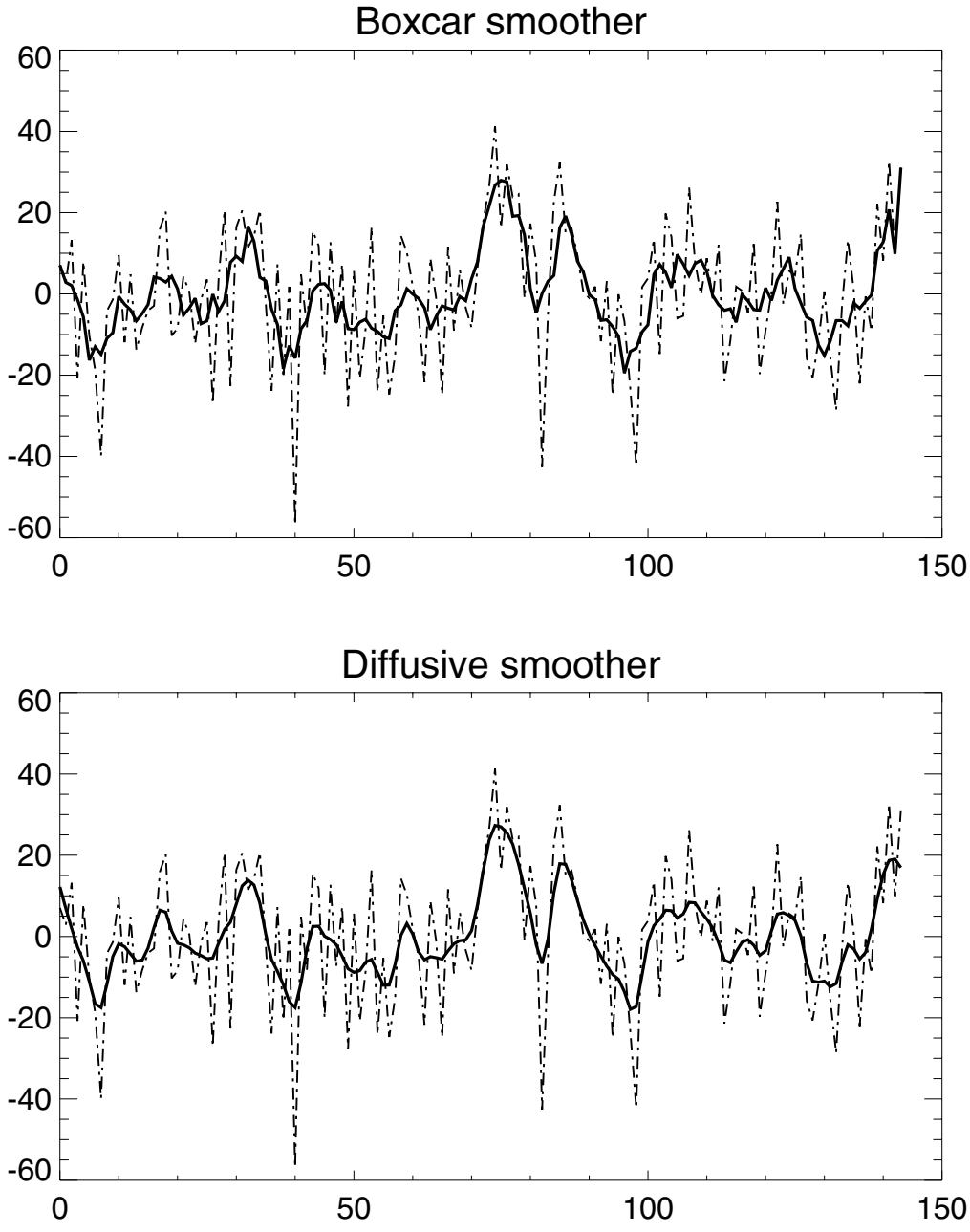

Figure 5: Top: original (dashed) and smoothed (solid) τ^x from the FSU data set, with a boxcar smoother of width 15. Bottom: with a diffusive smoother.

there is no guarantee that the average of the anomalies will be zero. In fact, they will easily average plus or minus half a degree. Forcing the HCM with anomalies which *actually* have a time average will not work; in particular, it will confuse the MOS corrector, which will try to remove the bias but be unable to do so (since it, too, is based on anomalies). The only thing which can be done is to reconstruct the original SSTs for the period which the HCM will be run over, and then form a new set of climatologies and and anomalies from that.

3.6. Constructing EOFs

The mechanics of constructing the EOFs from the smoothed observed data deserves a few comments. In addition to selecting the program to compute the EOFs, it must be decided whether to base them upon the correlation matrix or the covariance matrix (this is generally an option in the EOF calculating program). All the results shown here are based upon the covariance matrix.

In the Climate Research Division there are at least three routines available to calculate EOFs: Tony Tubbs' "SWEOF" program, Niklas Schneider's IDL routine, and my own program based upon LAPACK.

The programs can be tested by using a synthetic data set consisting of a plane traveling wave, which should result in exactly two EOFs, 90 degrees out of phase.

3.6.1. SWEOF routine

This routine calculates EOFs in a "switched" time-space decomposition, which is faster and uses less memory when there are more spatial points than temporal points (Preisendorfer 1988 , p. 64). The motivation for writing my own program was the discovery that the SWEOF routine has bugs which might result in errors in the calculated EOFs, and can certainly cause immediate program crashes and core dumps. I therefore recommend that this program not be used, unless only to verify the result of the other EOF routines. Additionally, it only runs in single precision, the code of the "householder" algorithm (AHOUSE.F) is extremely hard to follow, and it is slower than the double precision LAPACK-based routine. Giving this program the test data set results in a program crash — noise must be added to prevent this.

3.6.2. IDL routine

This is handy when you are already using IDL. On the other hand, it computes the EOFs using a SVD technique in *single precision only*, which is, in my opinion, questionable. Nevertheless, in practice it gave results essentially indistinguishable

from the LAPACK-based routine. Giving this routine the test data set yields the correct EOFs.

It perhaps isn't obvious that this routine should work, since it is based on the assumption that the SVD of the covariance matrix yields the eigenvalues and eigenvectors (at least, it wasn't obvious to me). Here is a proof, supplied by Elise Ralph via Niklas Schneider:

Given a symmetric matrix A (the covariance matrix):

$$A = A^T \quad \text{since symmetric} \tag{7}$$
$$A = usv^T \quad \text{definition of SVD} \tag{8}$$

where s has only diagonal elements. Thus,

$$\begin{aligned} usv^T &= A = A^T = vsu^T, && (9)\\ \Rightarrow usv^T &= vsu^T, && (10)\\ \Rightarrow v^T usv^T &= v^T vsu^T = su^T, && (11)\\ \Rightarrow v^T usv^T u &= su^T u = s, && (12)\\ \Rightarrow v^T u\, s\, v^T u &= s, && (13)\\ \Rightarrow v^T u &= 1. && (14) \end{aligned}$$

By the properties of SVD,

$$v^T v = 1, \tag{15}$$

therefore, if A is symmetric then we have

$$u = v. \tag{16}$$

Again by the definition of SVD,

$$\begin{aligned} A &= usv^T, && (17)\\ \Rightarrow Av &= us. && (18) \end{aligned}$$

Using Eq. 16,

$$Au = us. \tag{19}$$

Since s has only diagonal elements, u must be the eigenvectors and s must be the eigenvalues. Note that this result is based upon the assumed symmetry of A; therefore, the SVD technique will *not* generally yield the eigenvalues and eigenvectors of an arbitrary (non-symmetric) matrix.

3.6.3. LAPACK routine

The final program is based upon a LAPACK routine which calculates eigenvalues and eigenvectors of a symmetric matrix. Since all the "hard" work is hidden inside the LAPACK routine, this code is easy to follow. It runs in double precision faster than SWEOF does in single precision, and gives the correct EOFs for the plane wave test data. This routine automatically selects either a "switched" or "unswitched" time-space decomposition, based on whichever will be fastest to compute. In practice it has only been extensively tested in the "switched" mode, which is the one which applies when there are more spatial points than temporal points. This is almost always the case with ocean/atmosphere models and data. The source code for this routine is given in Appendix A.

3.7. Simultaneous EOFs

An important additional wrinkle in calculating the EOFs is that some of them should be calculated *simultaneously*. What this means is that rather than just calculating the EOFs of τ^x and τ^y separately, resulting in two sets of eigenvectors and two sets of principal components, the two data sets should instead be joined, at each month, into one *geographically larger* data set. The EOFs are then calculated on the joined data set, resulting in one set of eigenvectors and one set of principal components. The eigenvectors are then split up geographically in *exactly the same fashion* as the two data sets were joined to begin with; this results in two sets of eigen*vectors*, one for each original data set. There is still only one eigen*amplitude*, which applies to both sets of eigenvectors.

Upon first hearing about simultaneous EOFs, it seems a little weird that it doesn't matter how you geographically join the data sets into one larger data set. So, for example, if you start out with the τ^x and τ^y data sets, each of which has extent over longitude (X), latitude (Y), and time (T), then you could make a "wider" data set with two times the X extent and put the fields next to each other, or you could make a "taller" data set with two times the Y extent and put one atop the other. You could choose to intersperse them by interleaving rows or columns, or even select every other grid point from one data set and the in-between grid points from the other! What is important is that whatever method you use to combine the two original data sets in the first place, you must use the same method to split the resulting single eigenvectors into two separate eigenvectors, one for each data set. This joining and splitting works because the mathematics which determines the EOFs does not operate on any spatial information besides that contained at individual points; we look at the final EOF and try to interpret what the two-dimensional spatial pattern means, but the mathematics did not take anything two-dimensional

into account when forming the EOF.

A complication of simultaneous EOFs is the normalization of the input fields. Imagine, for example, simultaneously taking the EOFs of τ^x and τ^y anomalies. These two fields have the same units, so if one were measured in, say, N/m^2, then the other should be also; thus there is no problem with geographically joining these two fields and then ranking points *on the basis of their variance*. However, imagine a case where the two fields have different units, such as τ^x and SST anomaly. In such a case, a huge (three standard deviations) variation in τ^x might be 0.1 N/m^2, while a modest (one standard deviation) variation in SST might be 0.5°C. In this case, because the numerical value of the SST variation is larger than the numerical value of the τ^x variation, the EOF routine will assign more importance to the SST variation rather than to the more significant τ^x variation. Clearly, the way around this problem is to normalize such fields by the standard deviation before forming the EOF.

It is an interesting question as to how to do this normalization. There are several possible ways:

- Calculate the total standard deviation at all time and space points and divide by that.
- At each time point, calculate the variance over space and divide by that.
- At each space point, calculate the total variance over time and divide by that.

The "best" way probably depends on the application. For the MOS corrector runs described below, the third normalization method was used. The reasoning behind this was that the SST and sea level variances tend to be larger in the eastern Pacific, but we care more about the model having skill in the central Pacific. Therefore, methods one and two would reduce the variability in the place where we wanted to keep predictive skill. On the other hand, it is worrisome that anomalies at points away from the main band of HCM variability, points where the normalizing variance is quite small, will be multiplied by a large value before being incorporated into the MOS. This is amoelierated by the use of the skill-based HCM output mask; only places which feel the HCM's output winds are included. Nevertheless, it is a point to be aware of.

Be sure to understand that when using a normalization with the HCM or MOS, you must save the normalization factors and re-apply them to the instantaneous model fields before using the output of the HCM. So, for example, were you to normalize the wind, the values which previously had varied between about +/- 0.01 N m^2 will now vary between about +/- 1.0 standard deviations. Therefore,

were you to feed non-normalized winds to an HCM constructed with normalized winds, then the predicted values would have the wrong magnitudes.

3.7.1. Weighting by area

When working with geophysical data, it is also important to weight the input data by the gridbox area before calculating the EOFs. So, for example, if gridboxes have a size of 2° longitude by 2° latitude, then boxes near the equator will contain much more surface area than boxes near the poles. Consequently, they should be weighted more when forming the EOFs. This twist also arises when non-uniform gridspacing is used, such as in ocean models with higher resolution in the tropics. If the gridboxes in such a model are not weighted properly, the resulting EOFs will exaggerate the tropical patterns. As a matter of practical import this may or may not make any real difference, but since it is easy to weight by area, it should be done.

The proper procedure for weighting by area has been described by a number of people; the following is taken almost verbatim from a short note by Ben Santer. If $z(x, y, t)$ is the original data set which we want to form the EOFs of, then define:

$$\hat{z}(x, y, t) = z(x, y, t)\sqrt{w(x, y)}$$

where $w(x, y)$ is the areal weighting to apply. The weighting values w can be given in any units.

Compute the EOFs of $\hat{z}$ using one of the methods outlined above; call the result $\hat{e}(x, y, n)$, where n is the mode number. The LAPACK-based program is particularly convenient to use for this because if the input data is in netCDF format with axes properly filled out in degrees, then the areal weighting is done automatically.

Finally, compute the true EOFs $e(x, y, n)$ of the original data by:

$$e(x, y, n) = \hat{e}(x, y, n)/\sqrt{w(x, y)}.$$

4. Step by Step Procedure

This section gives the step by step procedure for constructing and cross validating an HCM. Although it is straightforward theoretically, in practice there are *many* little details which have to be remembered. Illustrations will also be supplied from HCM version 3.

1. Select observed data sets for SST, τ^x, and τ^y. This is covered above.

2. Choose the time period to use for forming the HCM and MOS corrector. Exactly what to pick here is a bit of an open question. The da Silva data sets go

back to 1945, but using only the period between 1965 and 1993 is probably safer. I often used the period 1979 to 1993 because it speeded up the process considerably.

2. Convert the observed data sets to monthly anomalies for the selected time period.

3. Smooth the monthly anomalies. A smoothing period of four to six months seems to be about right. Figure 6 shows original and smoothed SST anomalies for the da Silva data set over various regions in the equatorial Pacific. The El Niño events can easily be seen.

4. Form the EOFs of the observed smoothed monthly SST anomalies; this is the predictor field. It is important to only do this in the (limited) region where the statistical relationship between the SST anomalies and wind anomalies is strong. Since you don't know what this is until you are finished, it is a bit of a chicken-and-egg problem. I used a strip about the equator of +/- 15.5° of latitude. All predictor and predictand fields outside the chosen latitude band should be set to zero.

I generally calculated twelve EOF modes for all predictor and predictand fields. It is important to understand that this is *not* the same as the number of EOF modes to use when actually predicting things with the statistical model. The number of modes to use is covered in the next section.

Figures 7 to 14 show the first four EOFs and PCs of the da Silva SST anomalies, for both January and July.

5. Form the simultaneous EOFs of the observed smoothed monthly τ^x and τ^y anomalies; this is the predictand field. You need to form these EOFs simultaneously because they represent one physical quantity. Figures 15 to 22 show the EOFs and PCs of the first four (simultaneously calculated) modes for the wind anomalies, with data from da Silva's wind set.

6. Calculate the regression coefficient between each principal component of the predictor field and each principal component of the predictand field.

This completes the specification of the HCM.

5. Cross Validation

Before you can actually use the HCM to make predictions you must choose the number of predictor and predictand EOF modes to use. This isn't as easy as it might be, because if you use as many modes as there are data points (in time) to begin with, then you can always fit the observed data perfectly, but predictive skill will be terrible. So, for example, if you have twenty years of data and decompose into twenty modes, then you can exactly reproduce the observed data. But when you come to predict the twenty-first year, your twenty modes will be excessively "tuned" to the vagaries of the previous twenty years and will not do so well. Bottom

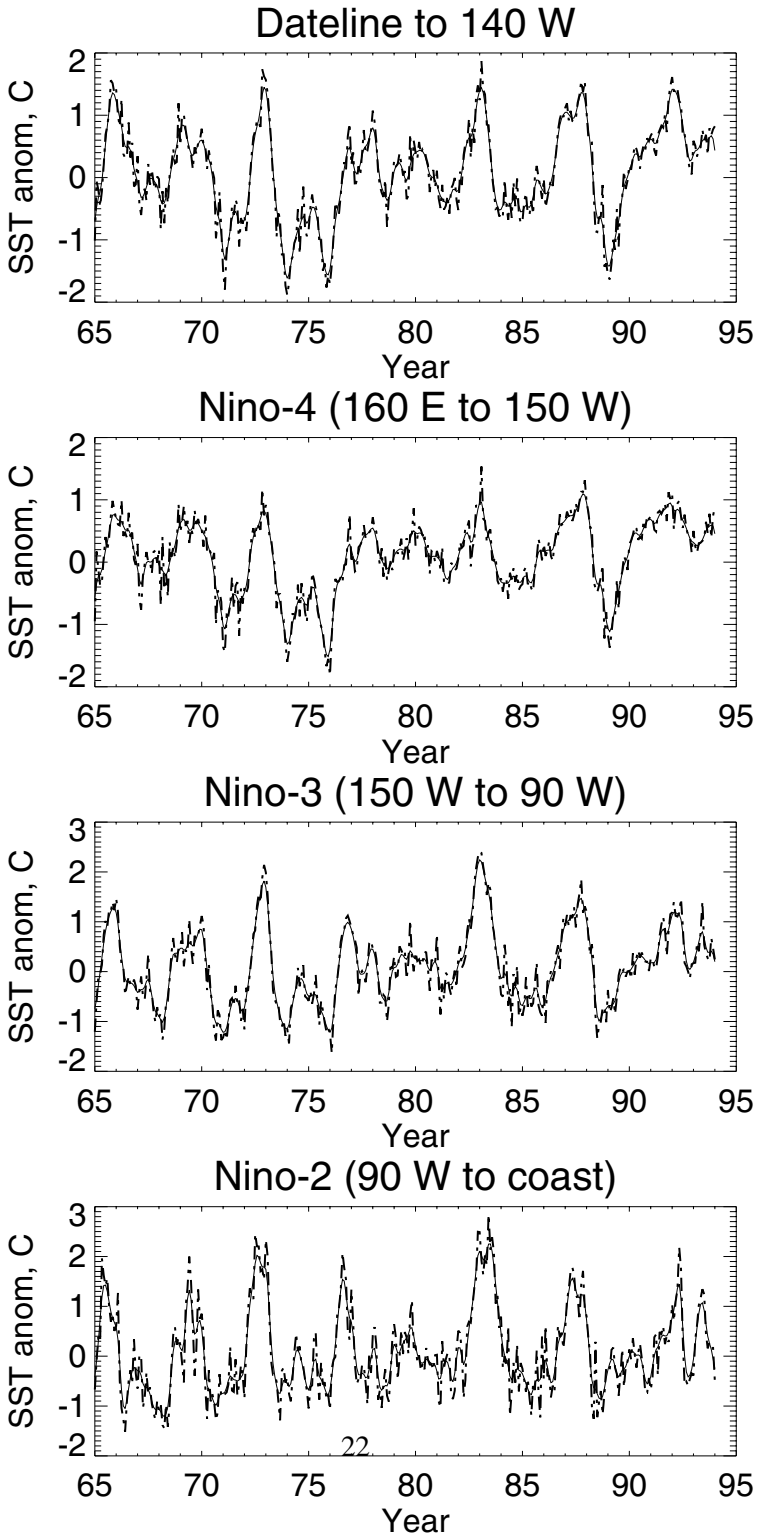

Figure 6: Original (dashed) and smoothed (solid) SST anomalies from the da Silva data set. All regions extend +/- 5°latitude.

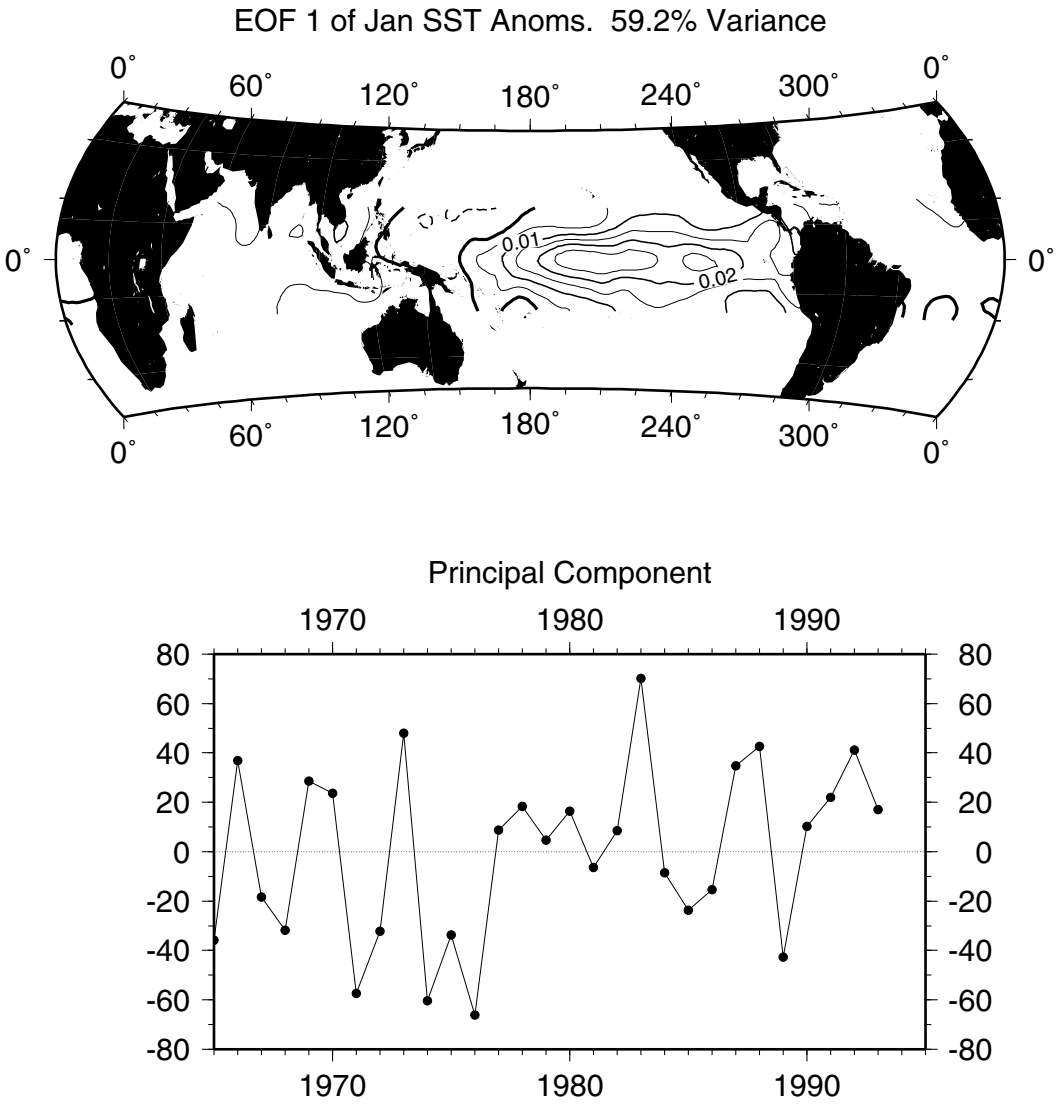

Figure 7: SST EOF #1 for January anomalies, calculated from da Silva's data set.

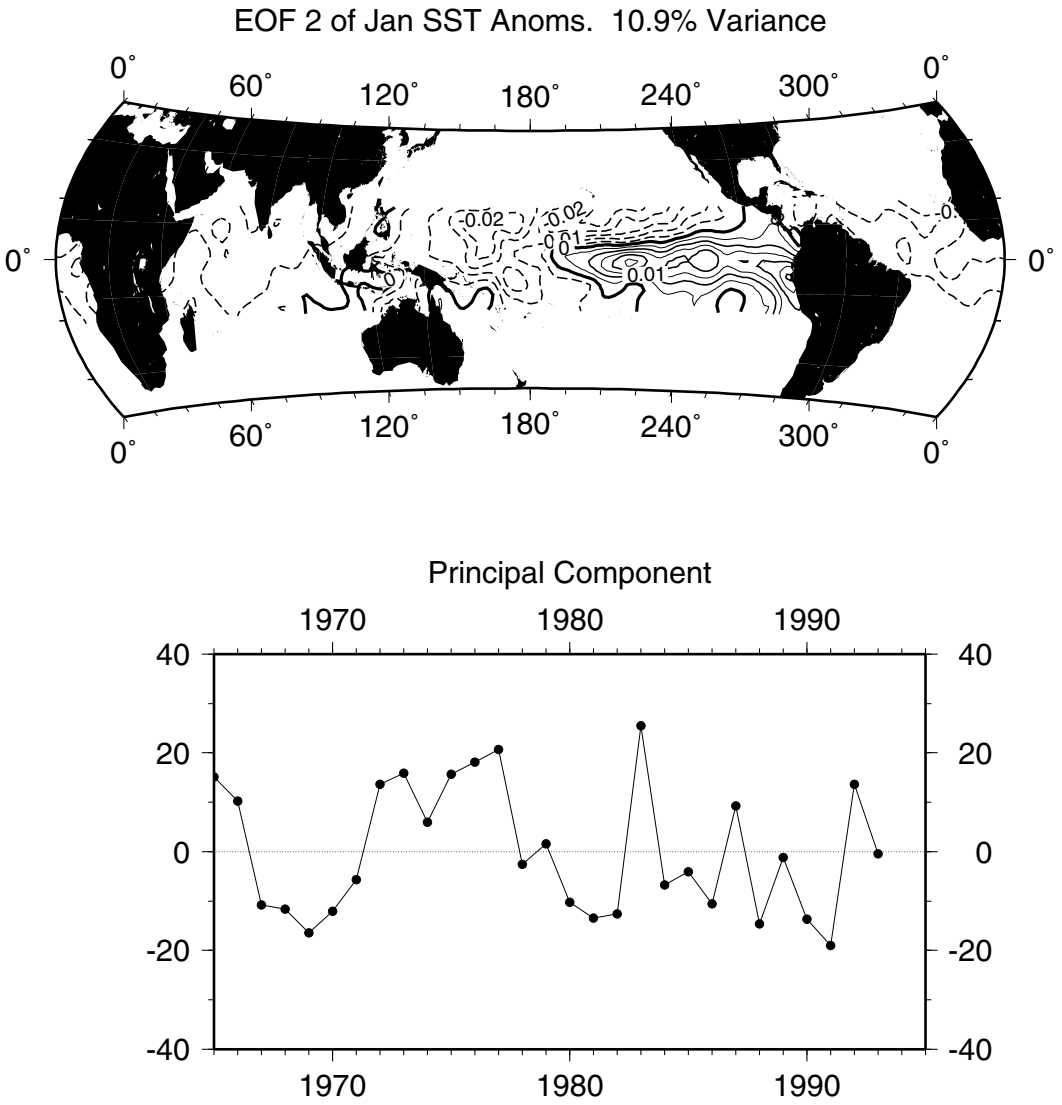

Figure 8: SST EOF #2 for January anomalies, calculated from da Silva's data set.

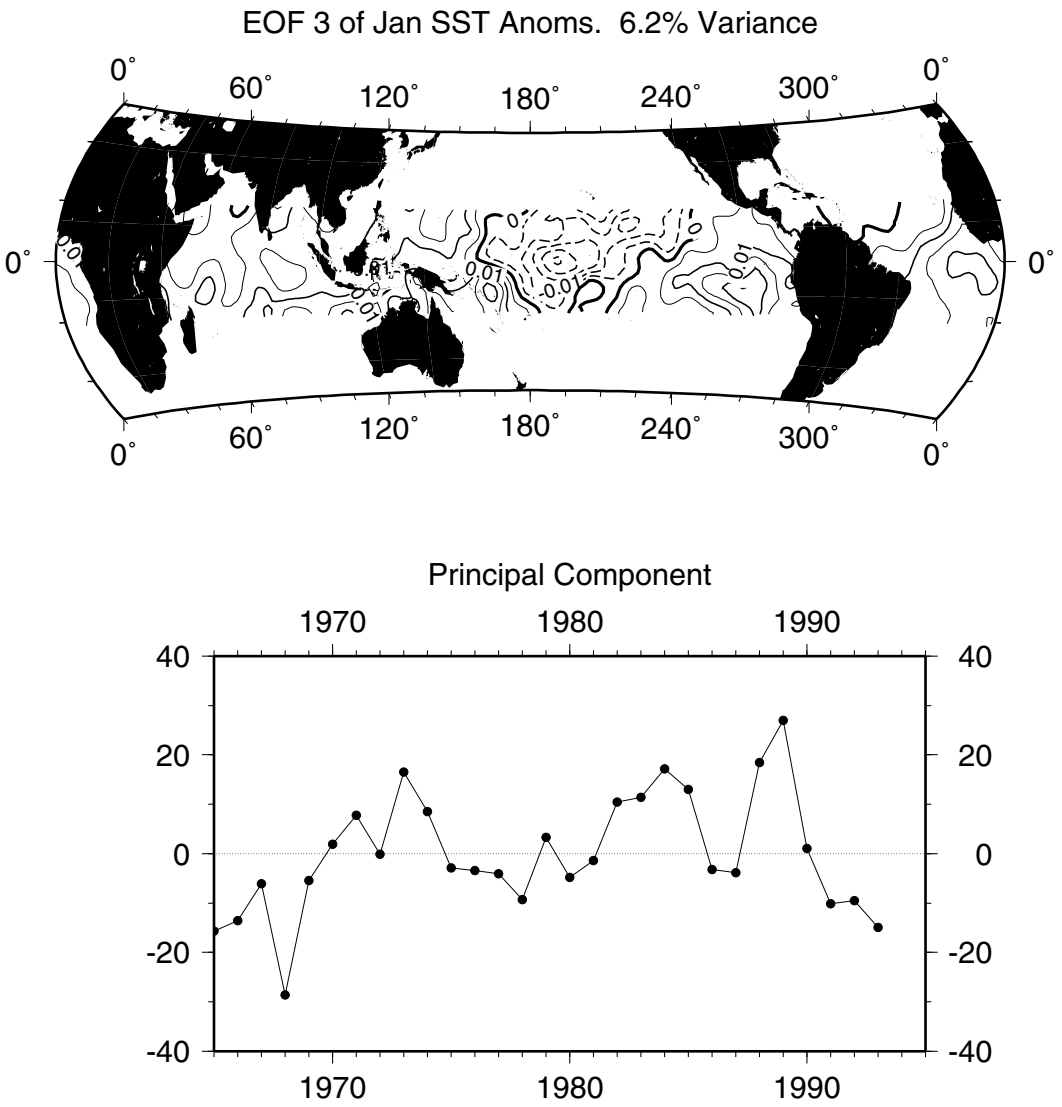

Figure 9: SST EOF #3 for January anomalies, calculated from da Silva's data set.

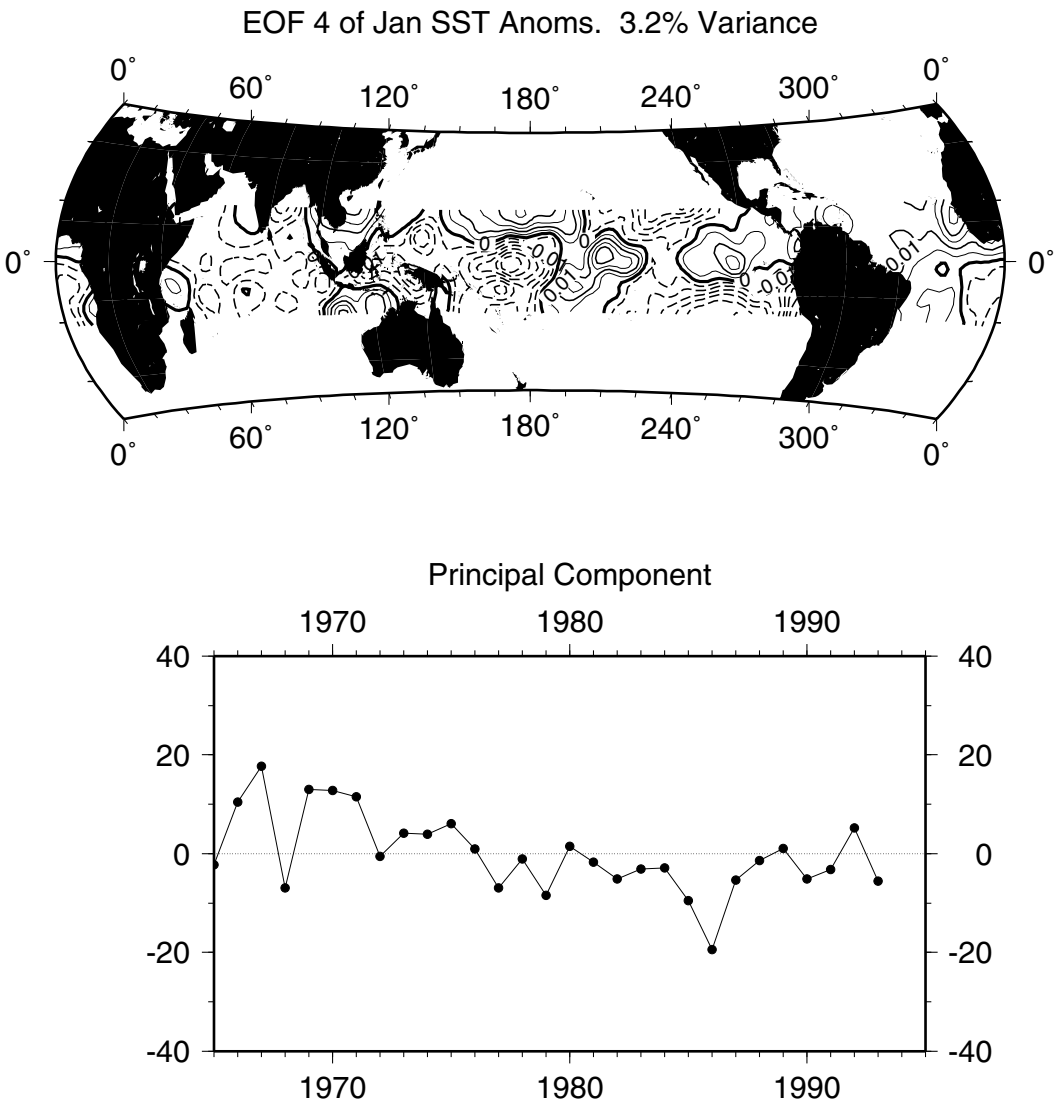

Figure 10: SST EOF #4 for January anomalies, calculated from da Silva's data set.

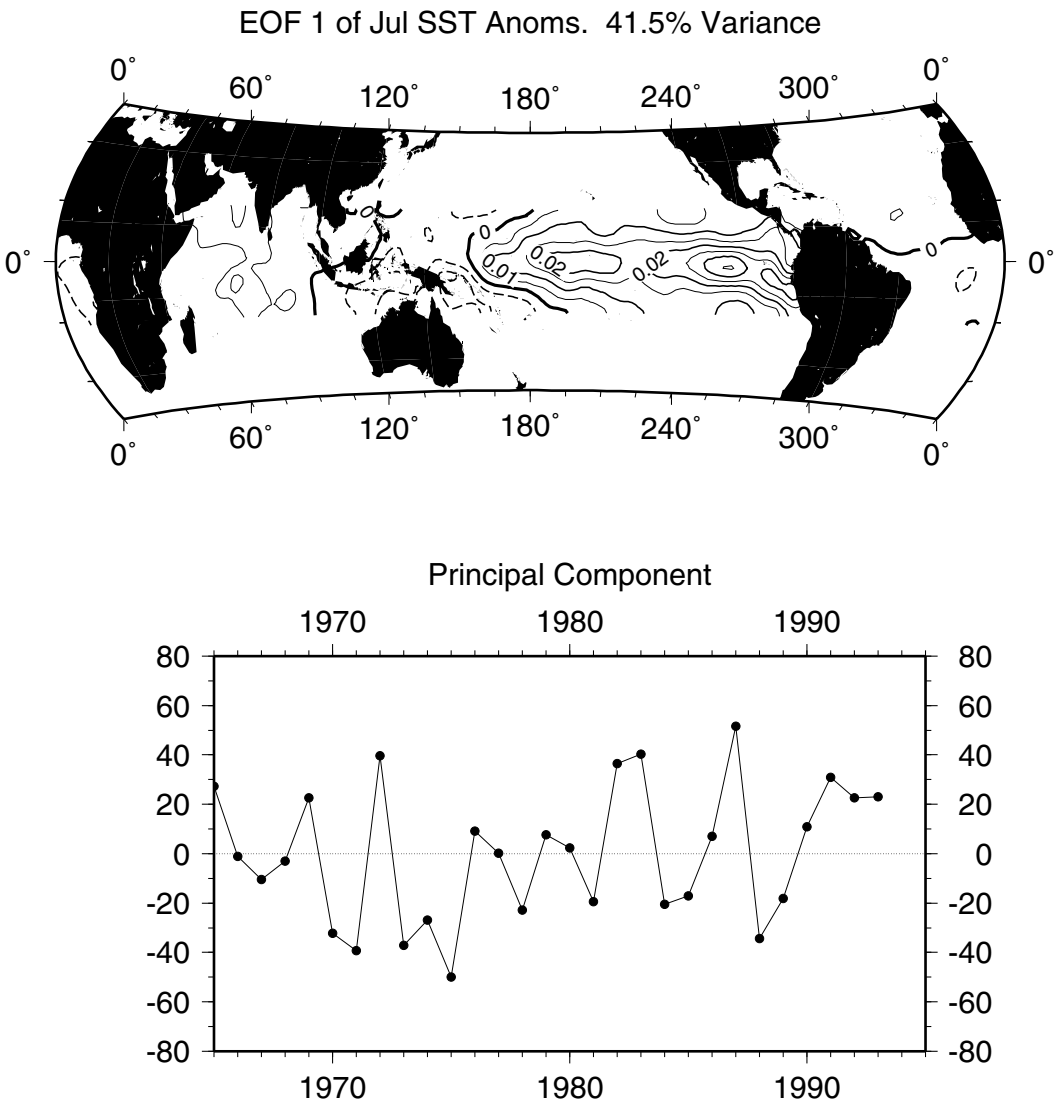

Figure 11: SST EOF #1 for July anomalies, calculated from da Silva's data set.

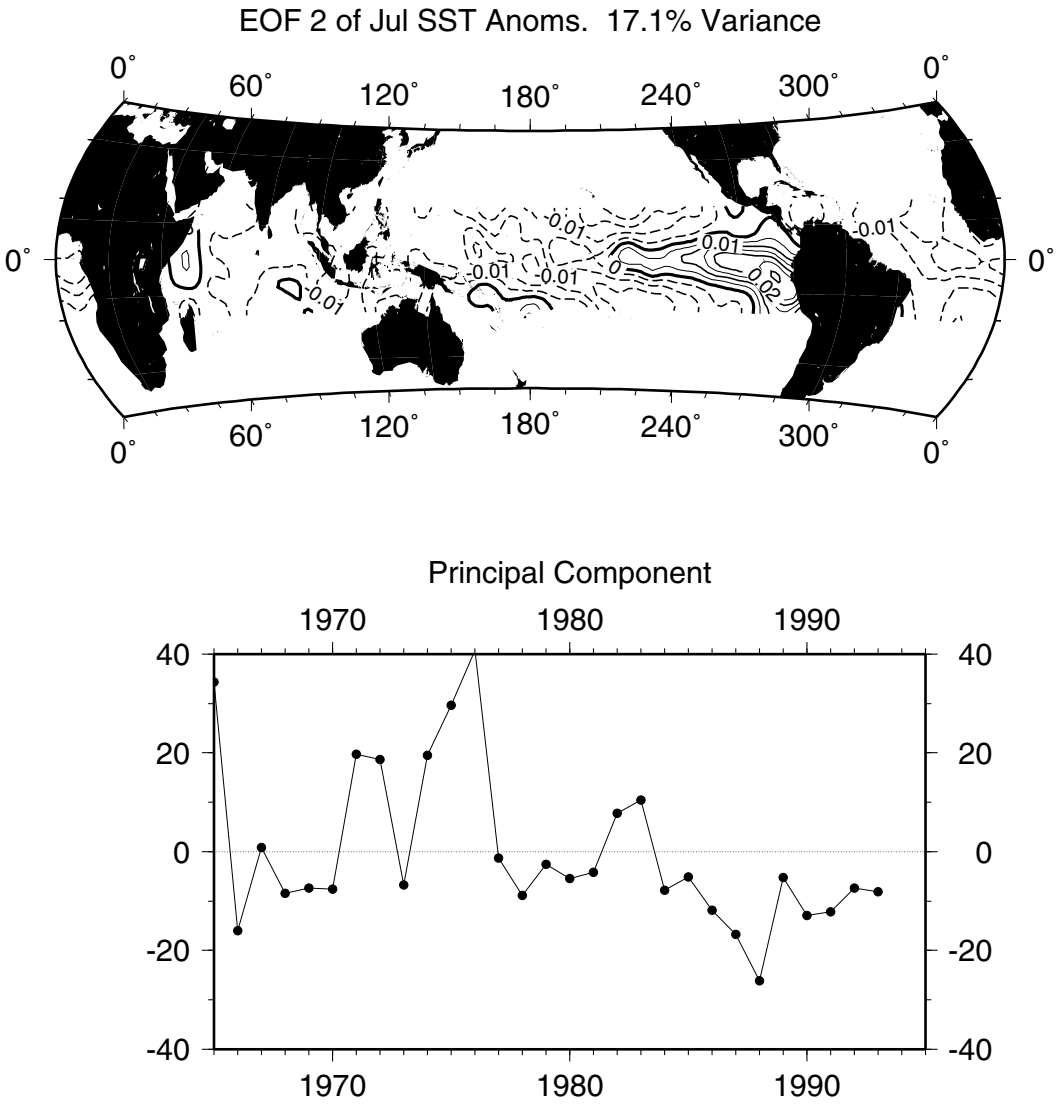

Figure 12: SST EOF #2 for July anomalies, calculated from da Silva's data set.

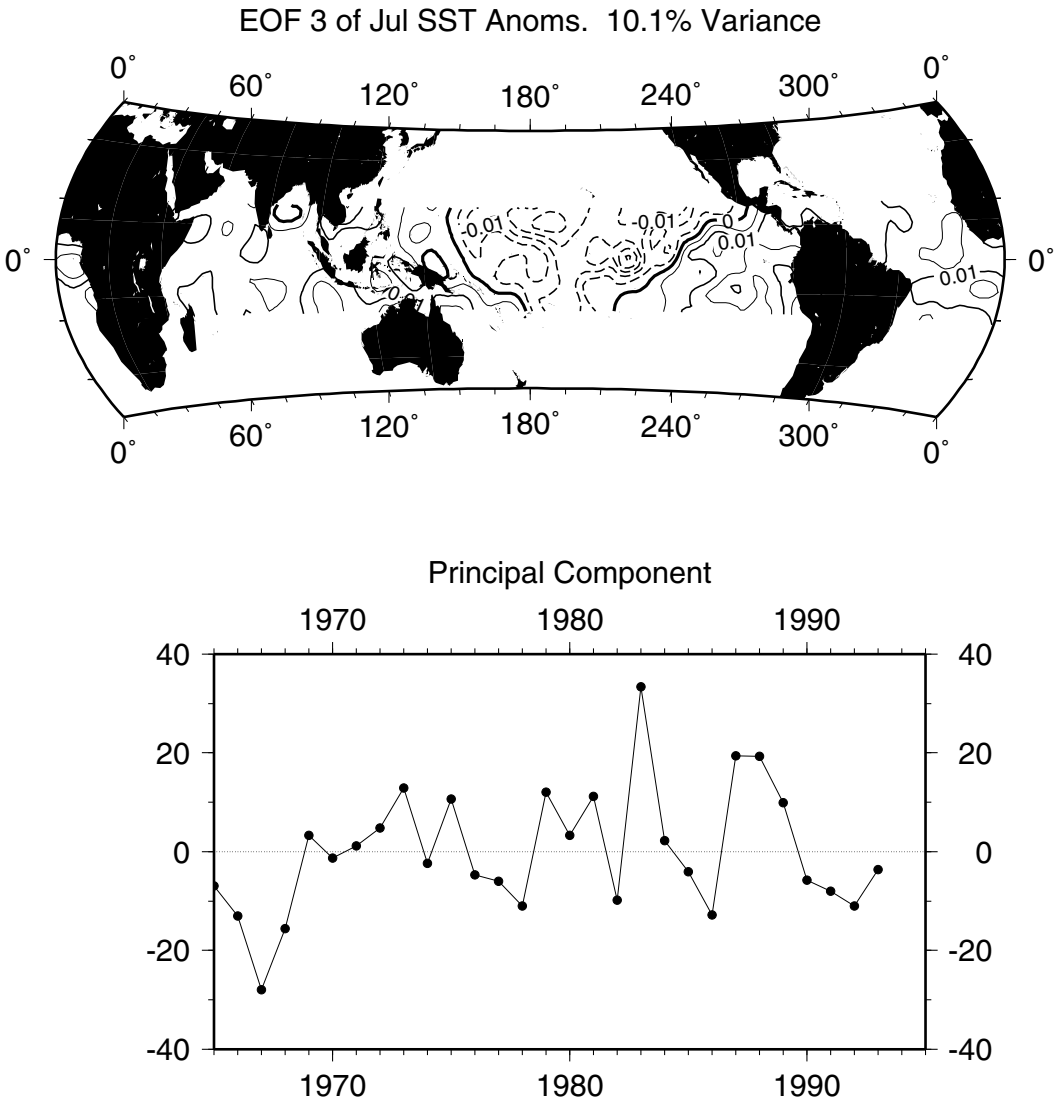

Figure 13: SST EOF #3 for July anomalies, calculated from da Silva's data set.

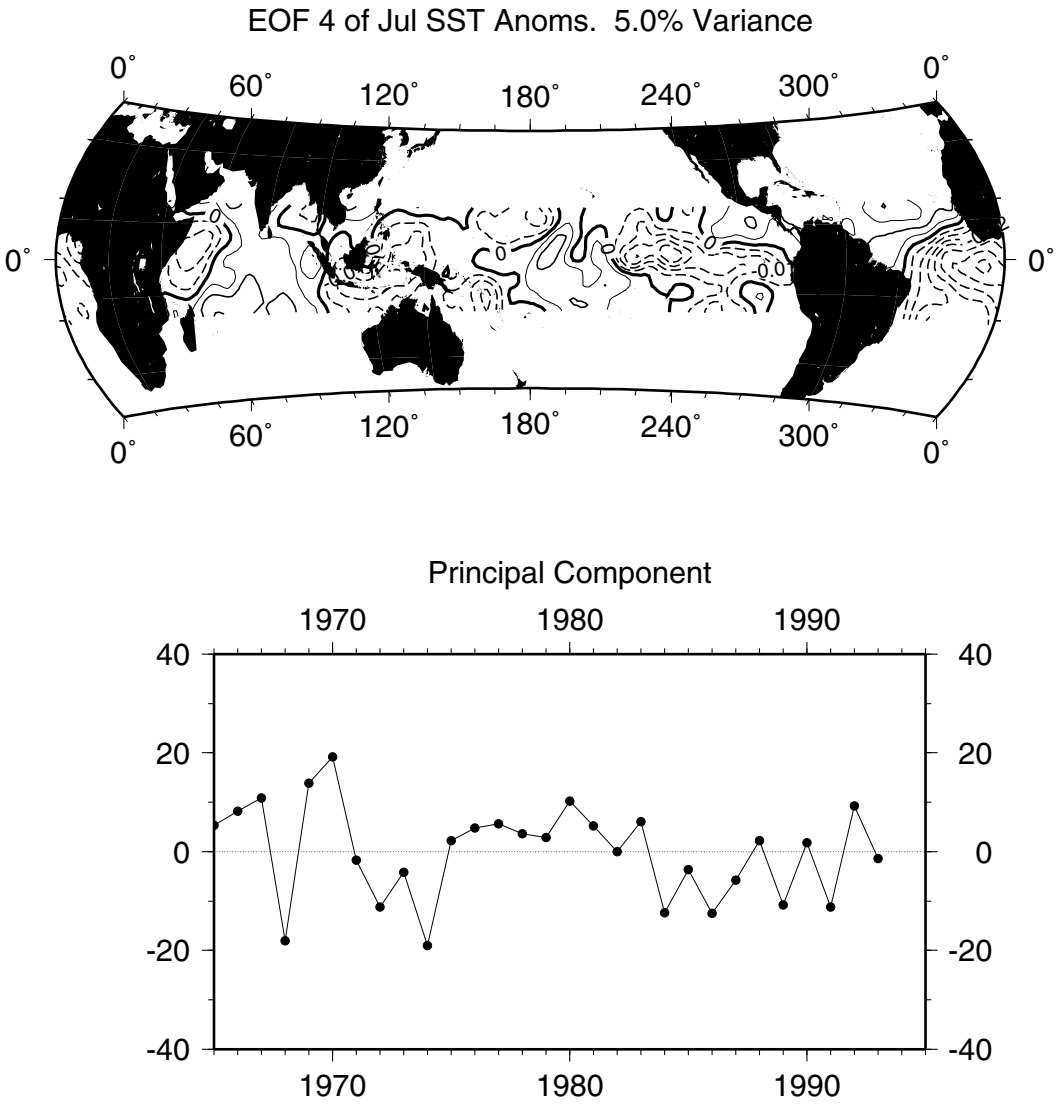

Figure 14: SST EOF #4 for July anomalies, calculated from da Silva's data set.

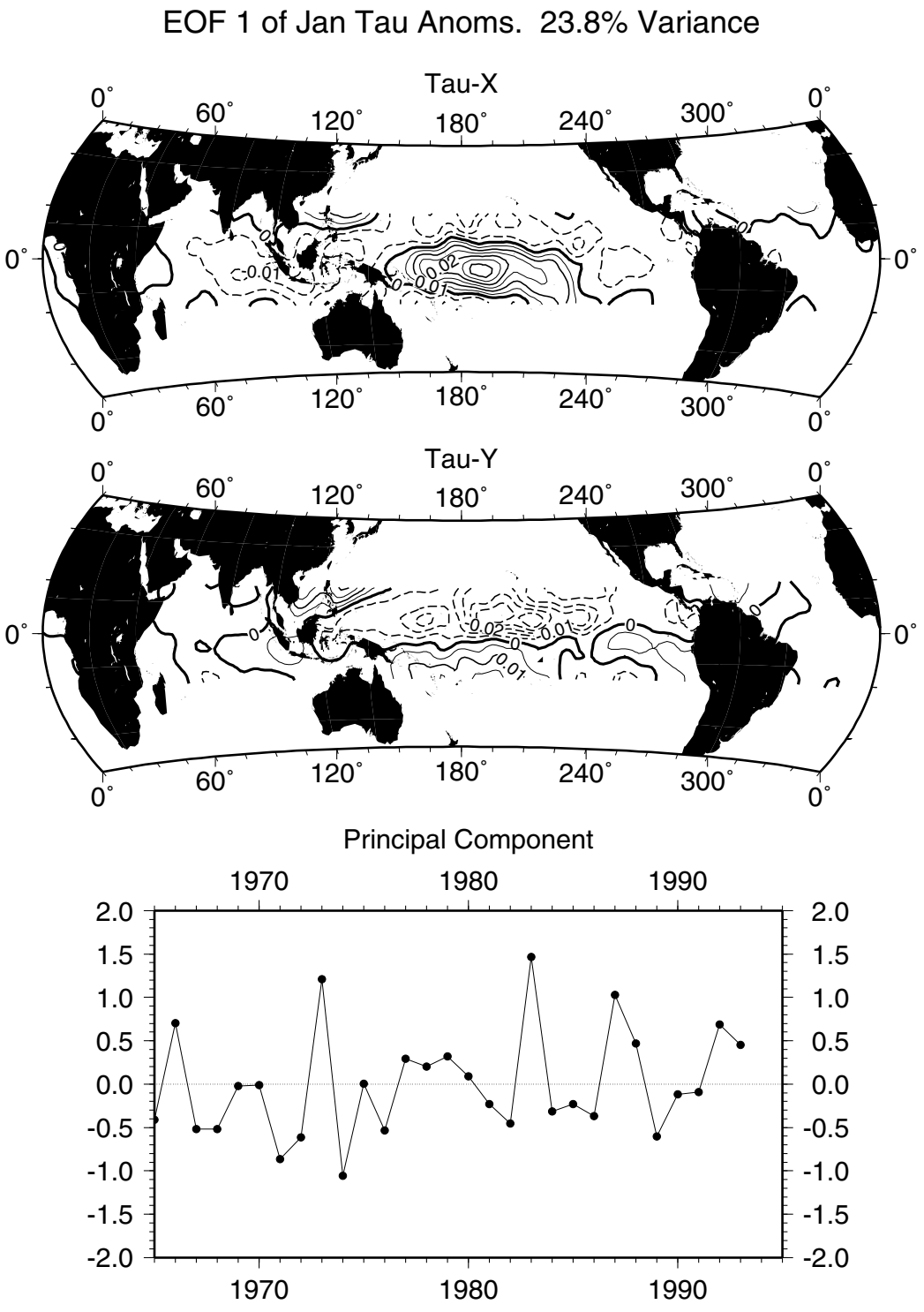

Figure 15: Tau EOF #1 for January anomalies, calculated from da Silva's data set.

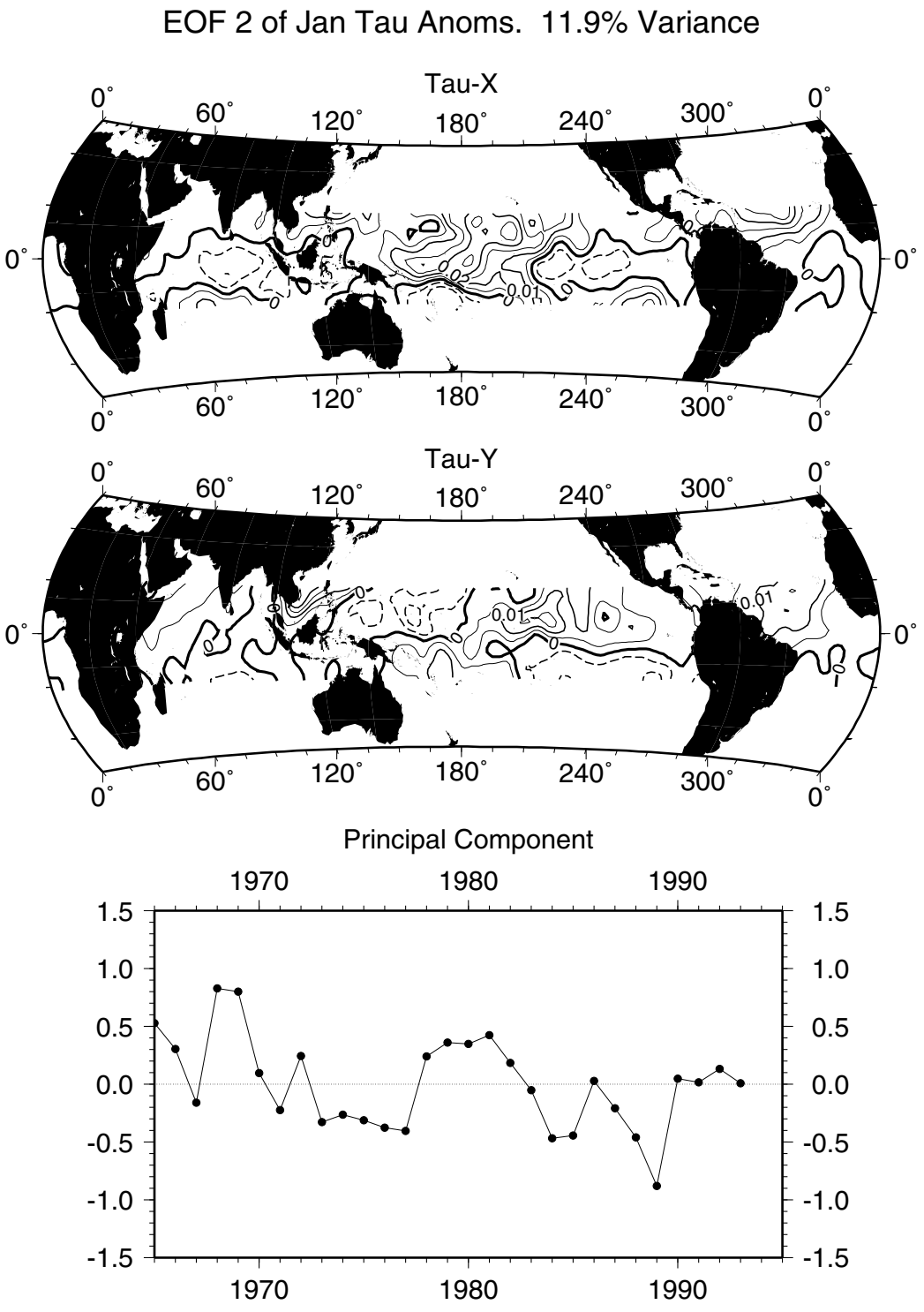

Figure 16: Tau EOF #2 for January anomalies, calculated from da Silva's data set.

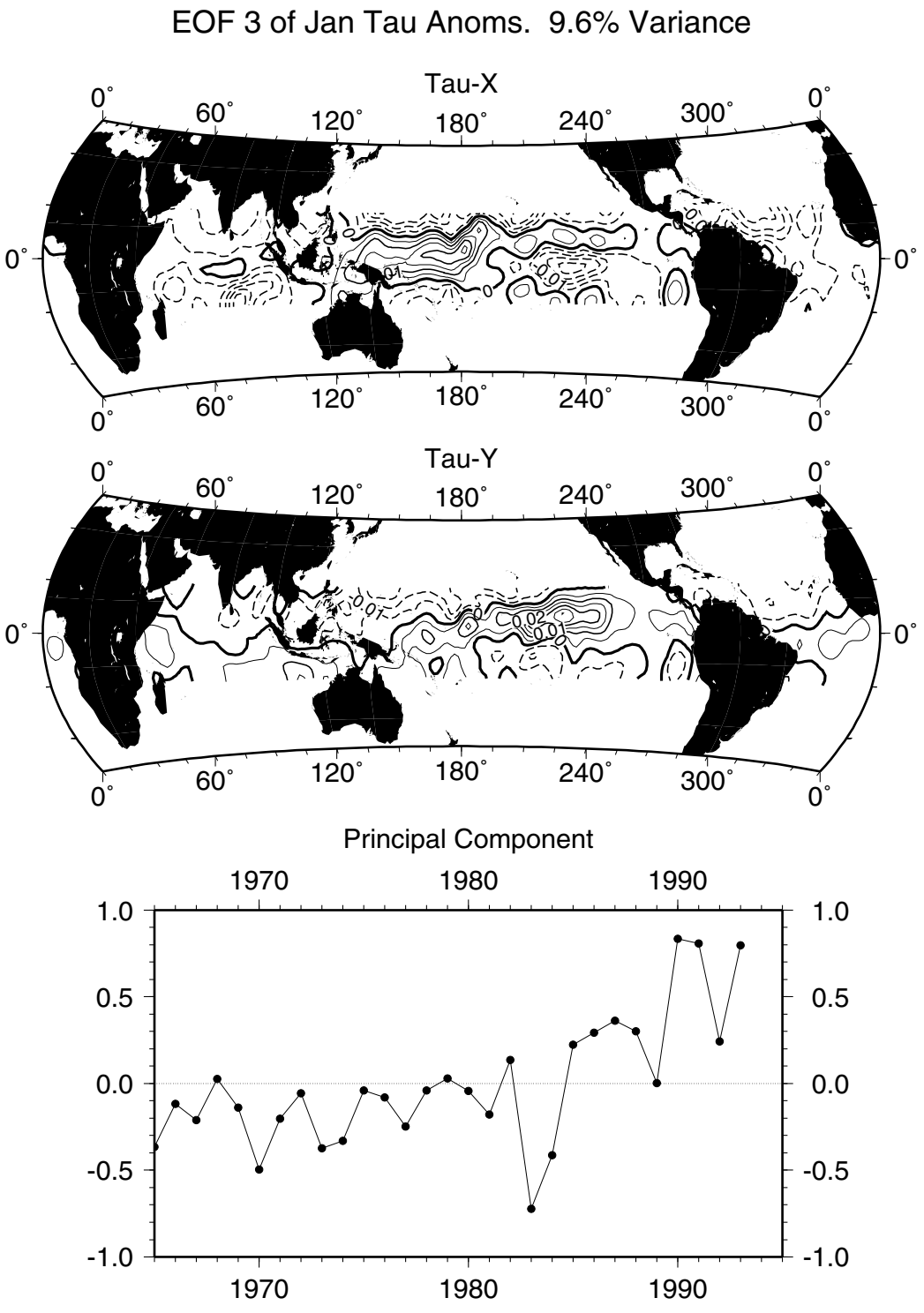

Figure 17: Tau EOF #3 for January anomalies, calculated from da Silva's data set.

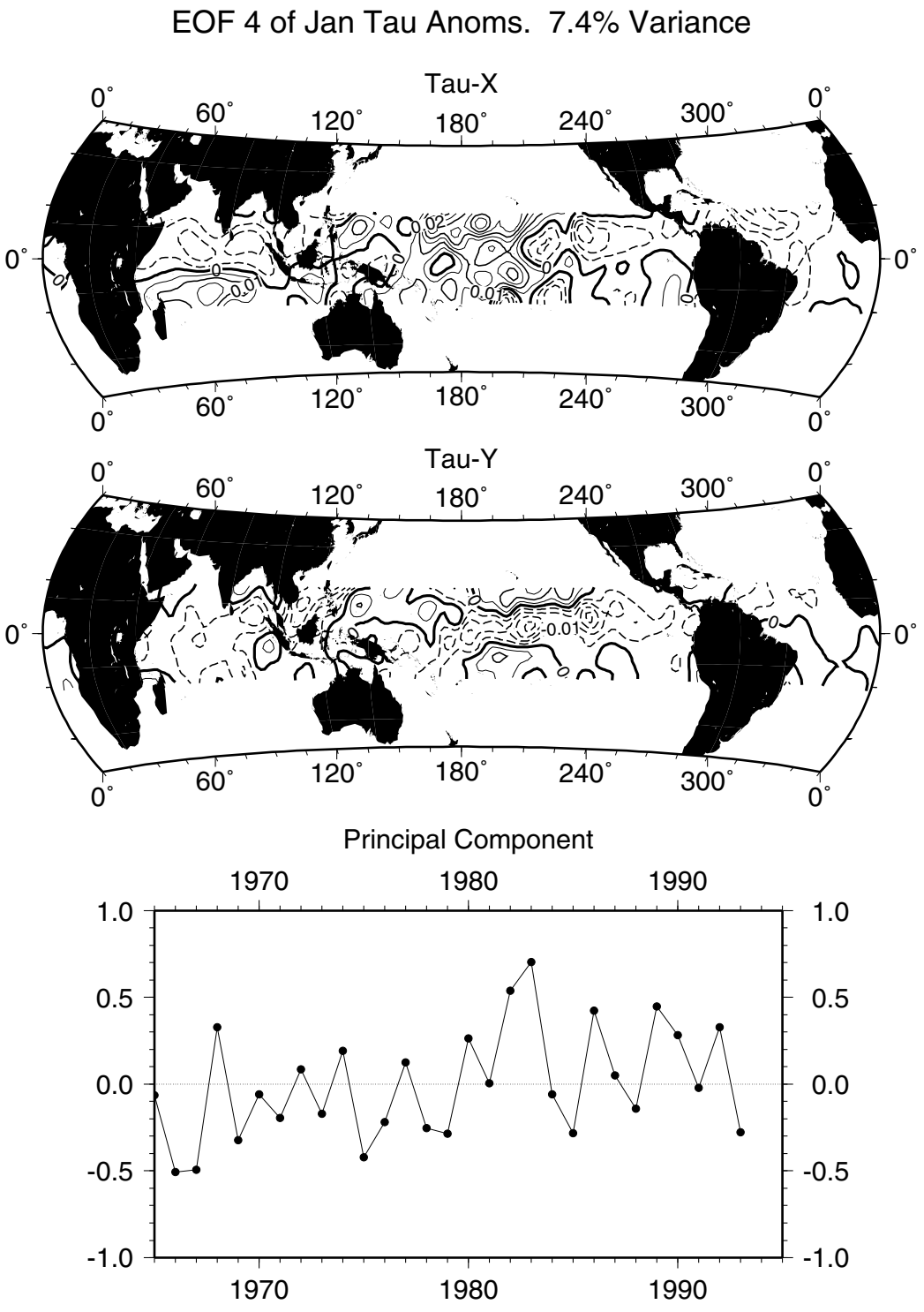

Figure 18: Tau EOF #4 for January anomalies, calculated from da Silva's data set.

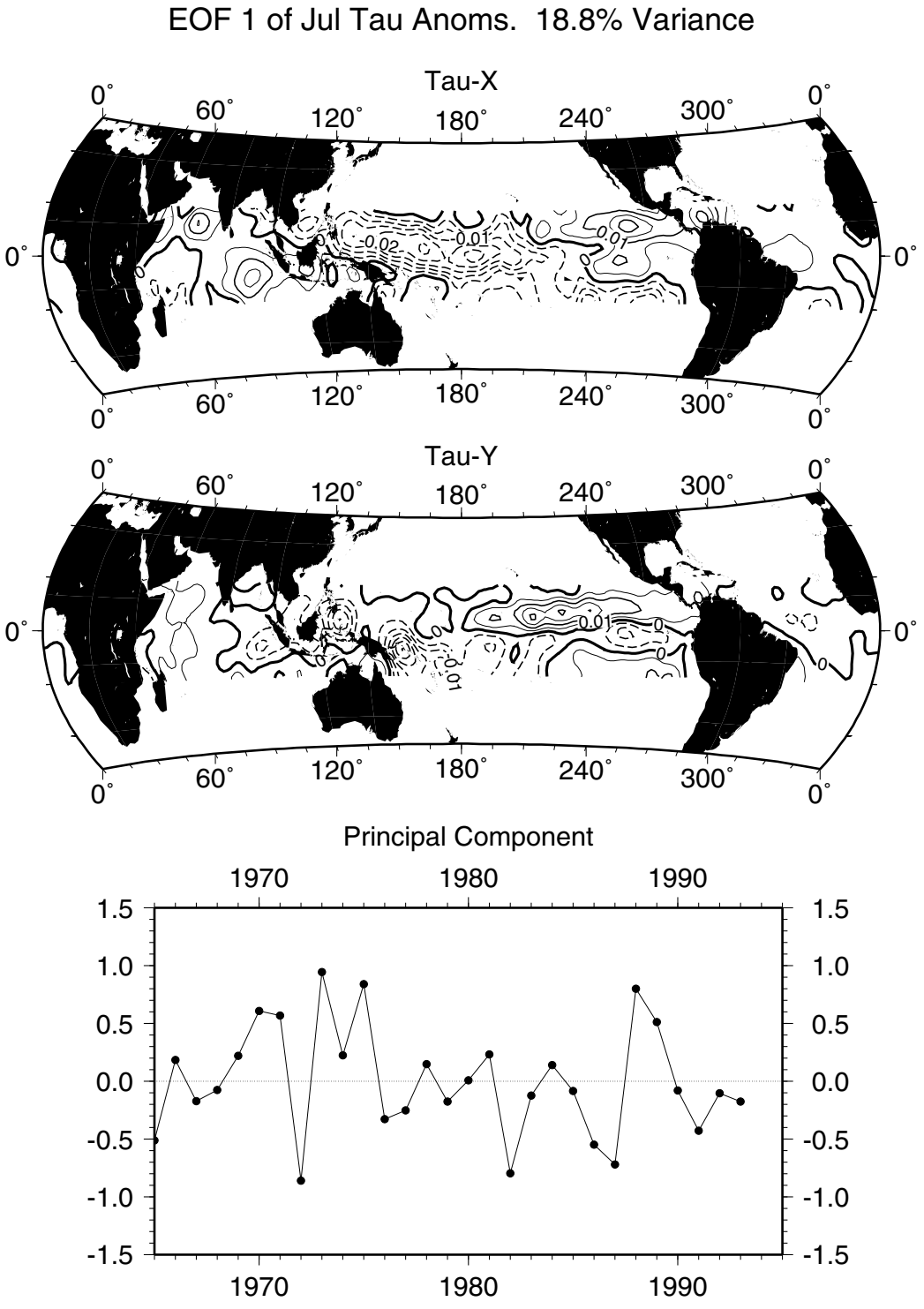

Figure 19: Tau EOF #1 for July anomalies, calculated from da Silva's data set.

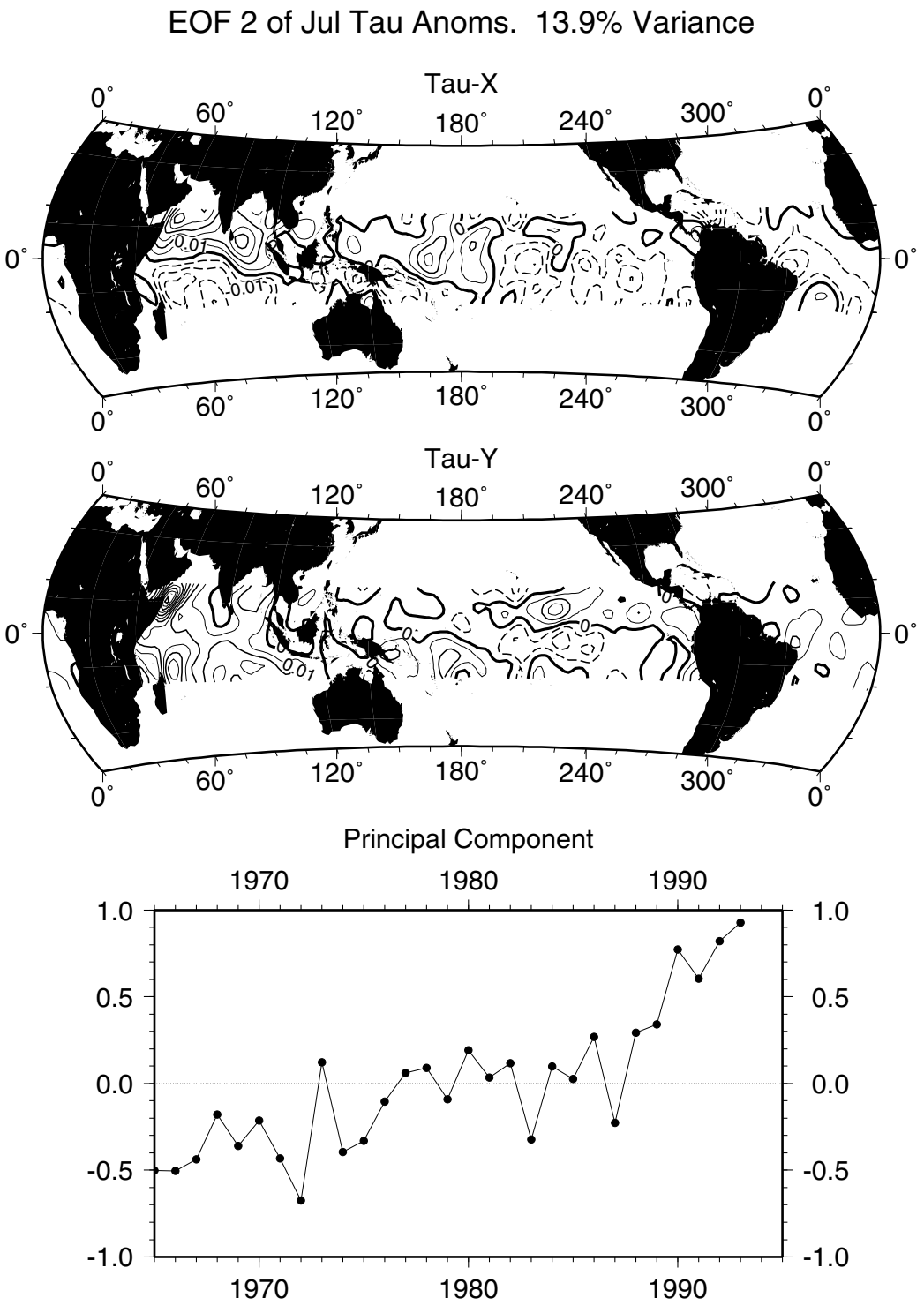

Figure 20: Tau EOF #2 for July anomalies, calculated from da Silva's data set.

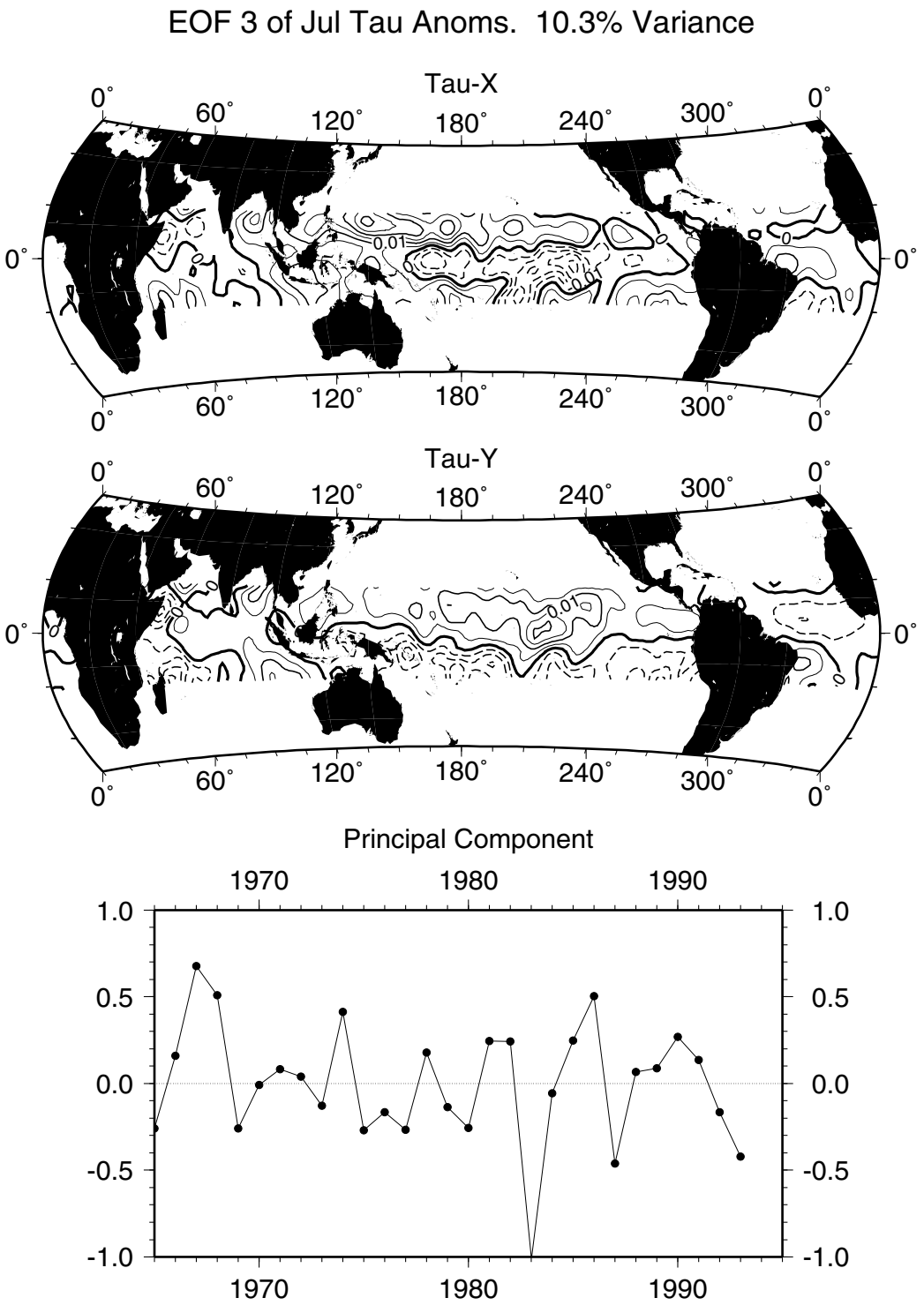

Figure 21: Tau EOF #3 for July anomalies, calculated from da Silva's data set.

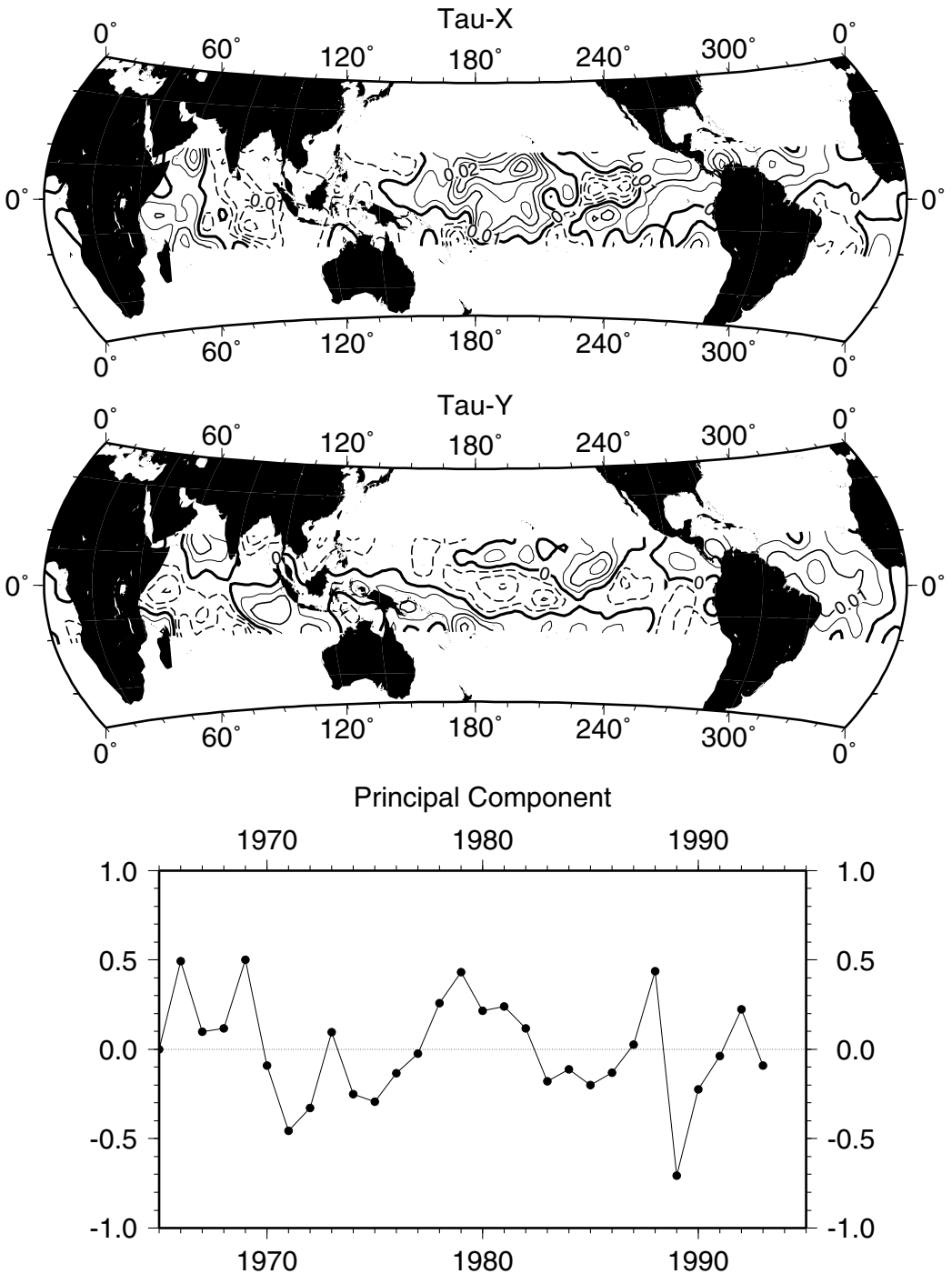

Figure 22: Tau EOF #4 for July anomalies, calculated from da Silva's data set.

line: you can't use too many modes. But you can't use too few modes either; if you do, you will be missing important parts of the variability.

An objective way to choose the number of modes to use for the predictor and predictand (they can, and usually will, be different) is as follows. Imagine that you start with, say, 26 years of observed data, 1965 to 1990. Construct 26 different complete HCMs—each one formed as outlined in the previous section—with each HCM made with all the observed data *except one year*. For example, the first HCM is formed by skipping year 1965, so includes years 1966 through 1990. Call this "HCM-skip_1965". The second HCM is formed by skipping year 1966, and so includes year 1965 as well as years 1967 through 1990; this would be "HCM-skip_1966". The result is 26 HCMs, "HCM-skip_1965" to "HCM-skip_1990".

You then have 26 HCMs whose predictive skill you can measure as a function of number of modes kept. Do this by forcing them with the observed predictor field *of the skipped year* and seeing how well the HCM reproduces the observed predictand field of the skipped year, while varying the number of predictor and predictand modes kept. Since no information from the skipped year went into construction of the HCM, this is an independent test. So, again considering our example with years 1965 through 1990, start by forcing HCM-skip_1965 with observed SST anomalies from year 1965, predicting τ^x and τ^y for 1965. But don't stop there; systematically vary the number of modes used in this prediction, for both the predictor and predictand. I typically examined from 2 to 18 modes in each, going by twos. Thus I had 81 separate predictions for 1965; 81 because there were 9 possibilities for the number of predictor modes (2, 4, 6, ..., 18), 9 possibilities for the number of predictand modes, and I took all combinations. Then use HCM-skip_1966 to make 81 predictions for 1966, and continue likewise through the rest of the 26 HCMs.

To evaluate the quality of these predictions, notice that you have, as a function of number of predictor and predictand modes kept, a complete time series of independent HCM predictions starting in year 1965 and ending in year 1990. You also, of course, have observed data for that same time period. So you can compare the time series of the predictions against the time series of observations, and see which number of kept modes gives the best results. Two ways to compare the time series are by simply correlating them at each point, and by computing the skill score at each point. If $o(t)$ is the observed time series (of monthly anomalies) at a point and $e(t)$ is the estimate, then the skill score s is:

$$s = 1 - \frac{\langle (o-e)^2 \rangle}{\langle o^2 \rangle}$$

Skill scores above 0.5 or so indicate the estimate is capturing most of the observed variability. Both the correlation and skill score methods are useful and should be

tried for comparison.

There are several complicating wrinkles in all this. One is that it makes sense to compare the predictions not to the actual (observed) data set, but rather to the so-called "reconstructed" data set. The reconstructed data set is that 2-D time series produced by multiplying the kept predictand EOFs by the kept predictand principal components, where the EOFs and principal components are calculated from the entire data set with *no* skipped years. The point of doing this is that using a limited number of EOF modes immediately throws away some of the variability, and there is no way you will be able to predict that discarded part. This cross-validation is not intended to examine how much variance you lose by using only a limited number of modes[1]; it is intended to see how good your *predictions* are as a function of number of modes. The best you could ever hope to do with your predictive scheme is capture the amount of variance which is in the kept number of modes, which in turn is less than the total amount of variance in the original data.

Another thing to consider is that the best number of kept predictor and predictand modes is likely to be different for different months. Don't make the mistake of only looking at one month's data when choosing the number of modes to keep. In particular, the winter and summer months may look very different.

6. Applying the HCM — the Fudge Factor

As is obvious if you think about it for a bit, the HCM's predicted wind anomalies can never be close in amplitude to the real ones. For one thing, you start out by smoothing the observed data, which will reduce the peaks right away. Then you only keep a certain number of EOFs, again throwing away some of the variance. Finally, the HCM will not be a perfect model — it will miss some of the variability. The result is that the final output wind anomalies from the HCM will be noticeably smaller than the observed anomalies.

The "solution" to this is to multiply the final output wind anomalies by a fudge factor (although for publication purposes it might better be referred to as a "variance-preserving renormalization constant"). Values in the range of 1.3 to 1.6 should be sufficient. If the final coupled model does not spontaneously generate measurable SST anomalies, increasing the fudge factor would be the first thing to try.

[1] you can get a feeling for how much variance you lose by using only a limited number of modes by summing the percentage of explained variance in the kept modes.

7. MOS Corrector

The previous sections describe how to construct a statistical model which uses input SST anomalies to predict τ^x and τ^y anomalies. However, there is a problem with applying such an HCM to a real ocean model: the ocean model may not generate realistic SST anomalies in the first place. In that case, the HCM's predictions will be worthless.

A MOS corrector tries to remove mistakes in the ocean model's calculated SST anomaly, so that the SST anomaly field can be successfully applied to the HCM. It does this by using data from a *hindcast* run, i.e., an ocean model run forced with observed atmospheric conditions, to see how well or poorly the model reproduces observed variability. If the model (and the observations!) were perfect, then the model's SST anomalies from the hindcast run would perfectly match observed SST anomalies over the same time period. In practice the two fields will be different, leaving us with two data sets: the incorrect model SST anomalies, and the (presumed) correct observed SST anomalies. The machinery developed in construction of the HCM can then be brought into play, and used to form a statistical model which takes as its predictor the model's SST anomaly field and has as its predictand the observed SST anomaly field. This is the MOS corrector. The MOS corrector works with monthly anomalies, just as the HCM does, and there is a separate set of MOS parameters for each month.

In practice, as always, there are several additional complications which must be taken care of. First, it is useful to include the model sea surface height anomaly field as a predictor in addition to model SST anomaly. Thus, in the MOS corrector, the predictor field EOFs are formed by taking the simultaneous EOFs of SST and sea surface height— there are two predictor data sets and one predictand data set for the MOS corrector. For the HCM, there was one predictor data set (SST anomaly) and two predictand data sets (τ^x and τ^y). However, there is a complication here which didn't arise previously: the normalization of the simultaneous EOFs which must be done when fields with dissimilar units are used. This was discussed previously, in the section on simultaneous EOFs. Note that in practical application, the normalization factors applied when constructing the simultaneous EOFs *must be kept* and applied during run-time to the model's instantaneously calculated SST and sea level anomalies before these are fed to the MOS corrector. Since there are 12 monthly sets of MOS parameters, there will be 12 normalization fields for SST anomaly and another 12 for sea level anomaly, one for each month.

The second twist is that the MOS corrector doesn't actually predict SST anomaly, it predicts SST anomaly *error*. So the above description wasn't actually true; what you really do is form the MOS corrector using (normalized!) model SST anomaly

and sea level height anomaly as predictor fields, and model SST anomaly error as the predictand field. Model SST anomaly error is defined as (model SST anomaly) - (observed SST anomaly). Once this error is predicted, you subtract it out from the model's SST anomaly field to produce an improved SST anomaly field (make sure not to add the error back in again!). The rationale for doing this is that a statistical model which predicted raw SST anomaly must necessarily reduce the variance of the output field (refer to the previous discussion of the "fudge factor"); this method avoids wholesale reduction of the final SST anomaly variance. I still applied a "fudge factor" to the output of the MOS corrector, for the same reasons as a fudge factor is applied to the output of the HCM, but it was modest—typically only 1.1.

7.1. Forming the MOS corrector: the Hindcast Run

There are a number of ways which the data for the MOS corrector could be formed, only one of which is correct. Recall that the data comes from a hindcast run of the ocean model. You might consider forcing the hindcast run with observed τ^x and τ^y anomalies, then using the model-generated SST anomalies as the basis for the MOS corrector. This is not correct. What you should actually do is use observed *SST* anomalies to force the HCM, which then generates the τ^x and τ^y anomalies used to force the model. The rationale for doing this is that it is more similar to the final state of the fully coupled model — i.e., it is using the HCM-generated wind fields. This ensures that the model climatology formed during this run, and later used as the basis for calculating the instantaneous model anomalies, is as close to the model climatology during the coupled run as possible.

A subtlety of the hindcast run is that the observed SST anomalies which are used to force the run *must* have a zero time average at every point. This means that you cannot use anomalies computed from any time period other than the one which the hindcast will be run over. In practice, this means that "observed" SST anomaly sets cannot be used directly; instead, the observed SST anomaly data set must be combined with the observed climatology set to form absolute temperatures, and then a new set of anomalies and climatologies calculated from this set of absolute SSTs for the period over which the hindcast will be run. It almost goes without saying, but note that the HCM must be formed from anomalies calculated for the *exact same* time period, otherwise it will not be able to function properly.

7.2. Non-locality and Gridpoint Separation

One of the things which makes the MOS corrector difficult to get working properly is the way that local effects can have global consequences. This arises because

the projection onto the predictor EOFs is basically a dot product over the entire domain of the currently observed predictor fields and the EOF fields. So imagine, for example, that just one model point is behaving oddly. If the EOFs have a non-zero value at that point, then the dot product of the EOFs over the entire domain will also have that odd behavior, even if every other point in the domain is working perfectly well. This non-locality makes the MOS very hard to debug; only one point can throw off everything everywhere.

The non-locality of the MOS has a practical aspect in the treatment of the western boundary regions. The western boundary currents are not well resolved by the HOPE model due to the very fine scale of those features. As a result, there is an appreciable difference in sea surface height between adjoining gridpoints, and, in particular, between adjacent even and odd gridpoints. As a result, the MOS predictions for the even and odd grids can be quite different in a way which depends on small details of the western boundary currents. This is not physically plausible, so to get around this problem HCM v3 masks the MOS predictor fields so that the western boundary regions are excluded.

7.3. Calculating Anomalies for the MOS

As described above, the MOS corrector is constructed on a monthly basis, with one set of MOS parameters for each month. A separate issue is how to calculate the instantaneous model SST and sea surface height anomalies which the MOS corrector requires. HCM v3 originally formed the anomalies by taking the difference between the instantaneous model fields and previously computed (from the hindcast run) monthly model climatology, with cubic interpolation between the monthly tabulated points. In practice this turned out to be deficient, for the reason shown in Figure 23. There are places where the sea surface height changes rapidly enough that one point per month cannot accurately capture the model's variability. As a result, trying to form instantaneous anomalies based on monthly climatology is not practical. Because of this, HCM v3 uses 5-day model climatology to form the instantaneous anomaly fields.

8. Results

There are many model parameters which influence the results, such as the (previously mentioned) fudge factors applied to the HCM's output winds and to the MOS corrector. Another sensitive parameter is the thermal coupling coefficient used to relax the model to observed SST climatology. In the standard HOPE model this is set to 40W m^{-2}, a typical value. However this is probably not a good value

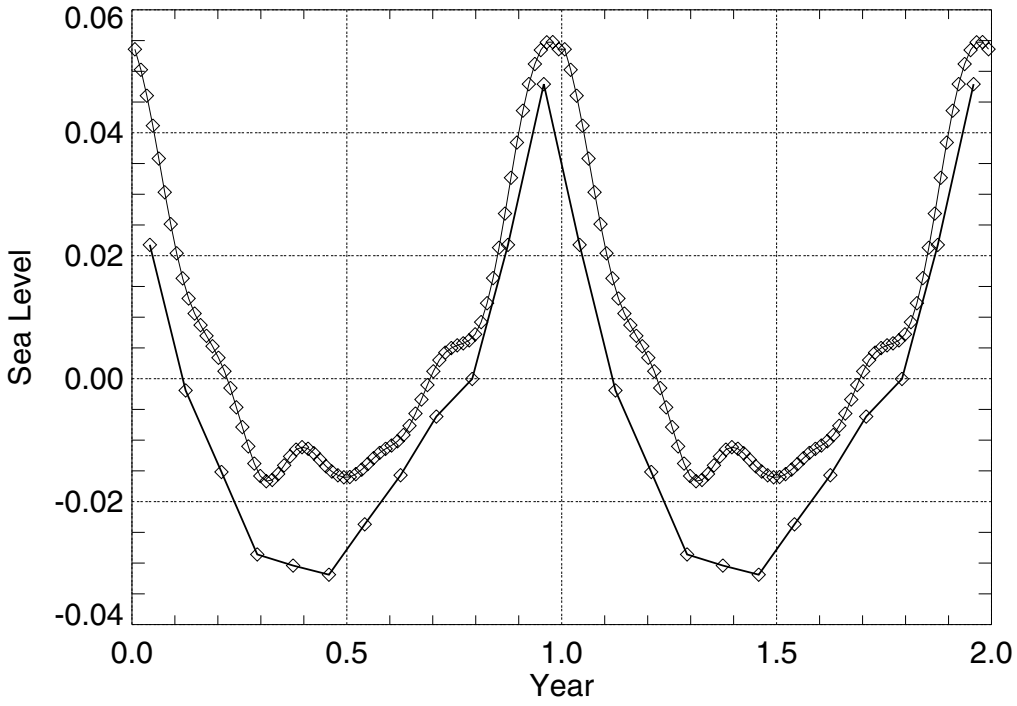

Figure 23: Annual cycle of model sea surface height for a point in the north central Pacific, as represented by samples taken at either one month or five day intervals. Curves are offset by 0.02 m.

to use. Schopf (1983) first found that a good representation of Pacific equatorial thermal variability required a substantially small value for the thermal coupling coefficient. In a coupled ocean/atmosphere model study, Barnett et al. (1991) found that a value of closer to 10W m^{-2} would be more appropriate for the eastern Pacific past the date line, increasing to 40W m^{-2} in the warm pool. Although HCM v3 gave reasonable results using a thermal coupling coefficient of 40W m^{-2}, the results are much better (particularly in the eastern Pacific) if this is decreased to 15W m^{-2}. Figure 24 shows observed SST anomalies over the period 1965 to 1994 in the equatorial Pacific, along with HCM v3 hindcast SST anomalies both with and without MOS correction. As can be seen, the agreement is good; the specific shapes of many events are captured in detail, although the extreme peaks tend to be truncated. All in all, the model seems to do a fairly good job in hindcasting. The SST-anomaly skill score of this hindcast is shown in Figure 25.

Examples of model predictions at lead times of 3, 6, 9, 12, and 15 months are shown in Figures 26 to 30. The results are quite gratifying for this particular time period, but it should be emphasized that not all times are captured so well. In particular, events where the hindcast is noticeably time-lagged behind observations, such as the 1972-1973 El Nino, do not seem to be forecast very well.

The entire 1965 to 1996 period forecast at a lead time of six months is shown in Figure 31. It is interesting to note that the period 1980 to 1996 yields better predictions than 1965 to 1980. This apparently is characteristic of many, if not all, models which attempt to predict El Nino; however, the reason for this is not clear.

9. Future Directions

Statistical models relate one set of data to another set of data in a way such that the second can be predicted from the first. In the HCM, this means that SST is used to predict surface winds; but the idea is quite general, as illustrated by the fact that the MOS corrector uses the model-predicted SST field to predict errors in the model SST field. However, there is no reason why this process cannot be continued; in the future, HCM v3 will have a component which uses model-predicted *predictions* as the predictor for actual predictions. The way this works is that the HCM will be used to make an entire series of predictions of previous years; since what actually happened in those years is known, a statistical model can be constructed relating those two data sets. Ideally, this would result in a model whose predictive power is increased, based on observations of how it has been systematically unskillful in the past.

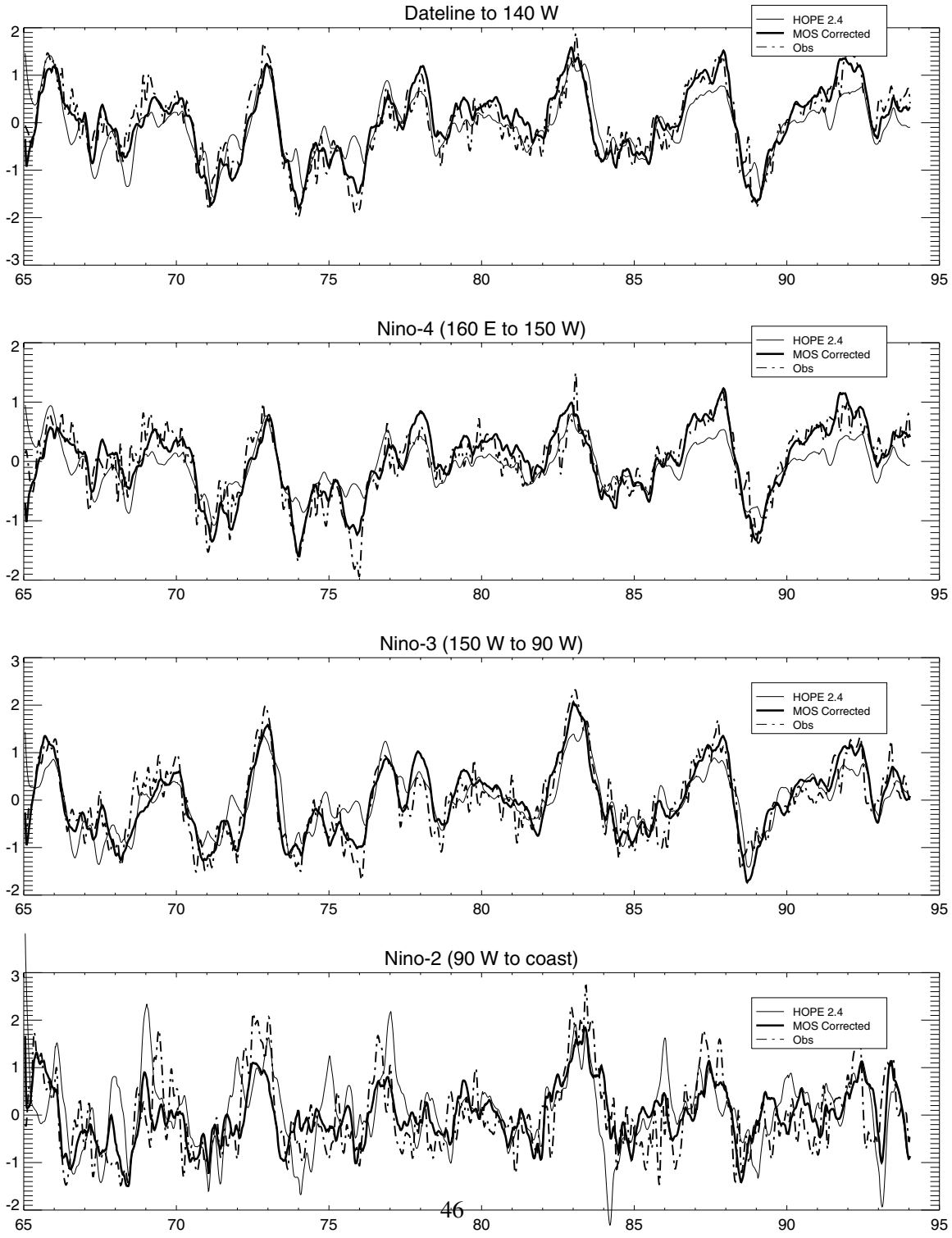

Figure 24: HCM v3 hindcast.

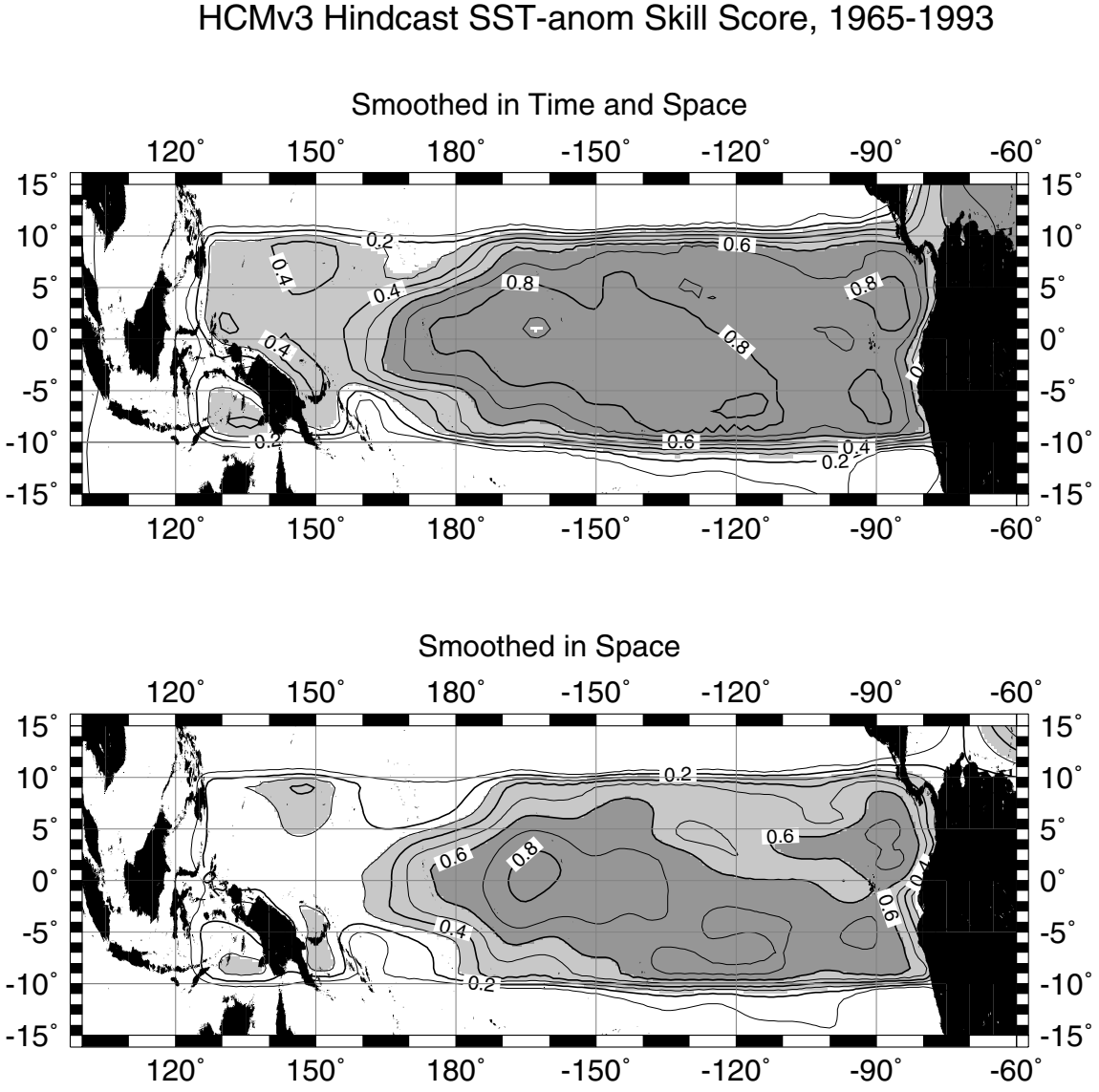

Figure 25: Hindcast SST skill score.

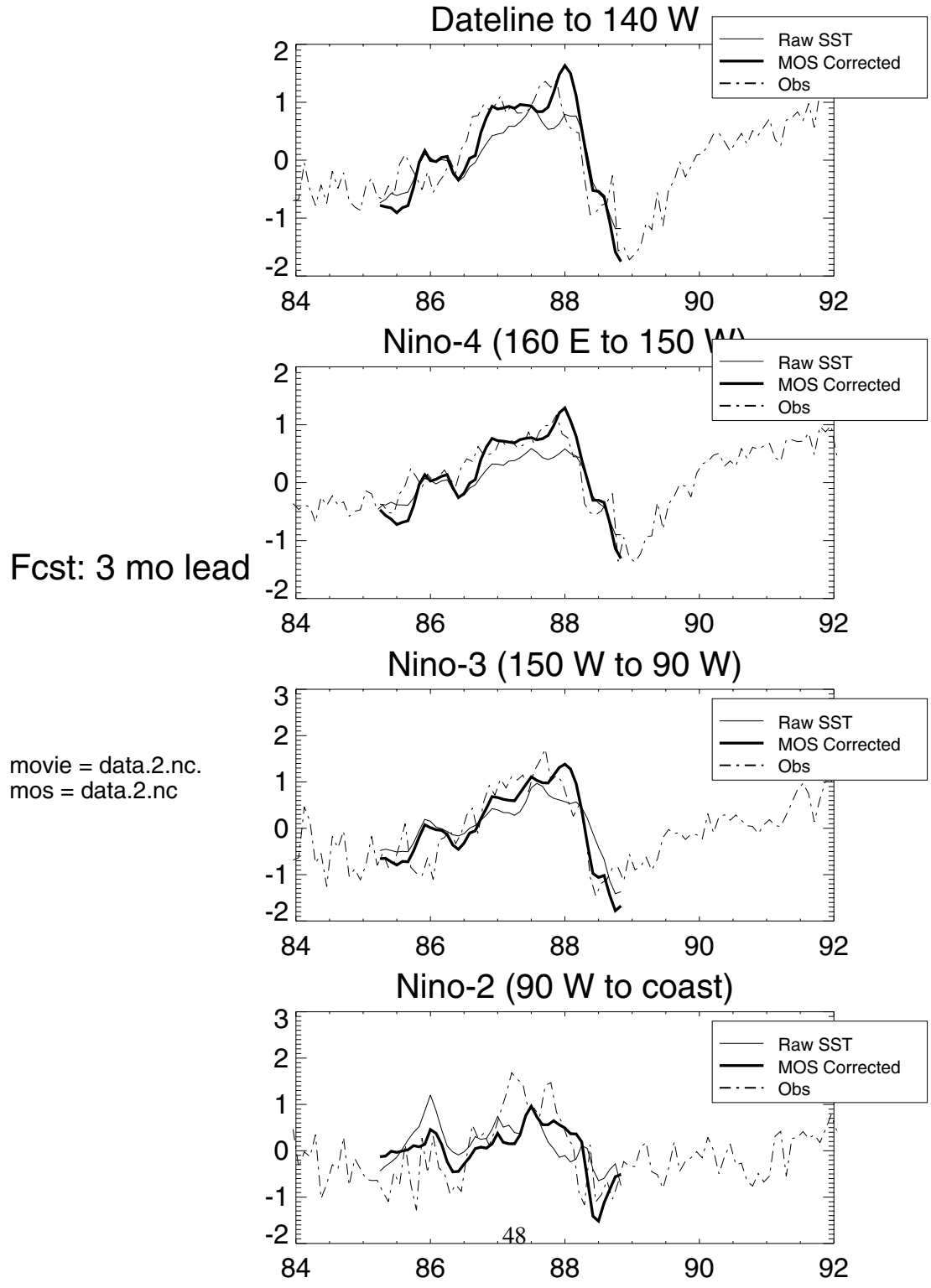

Figure 26: Forecast at 3 month lead time.

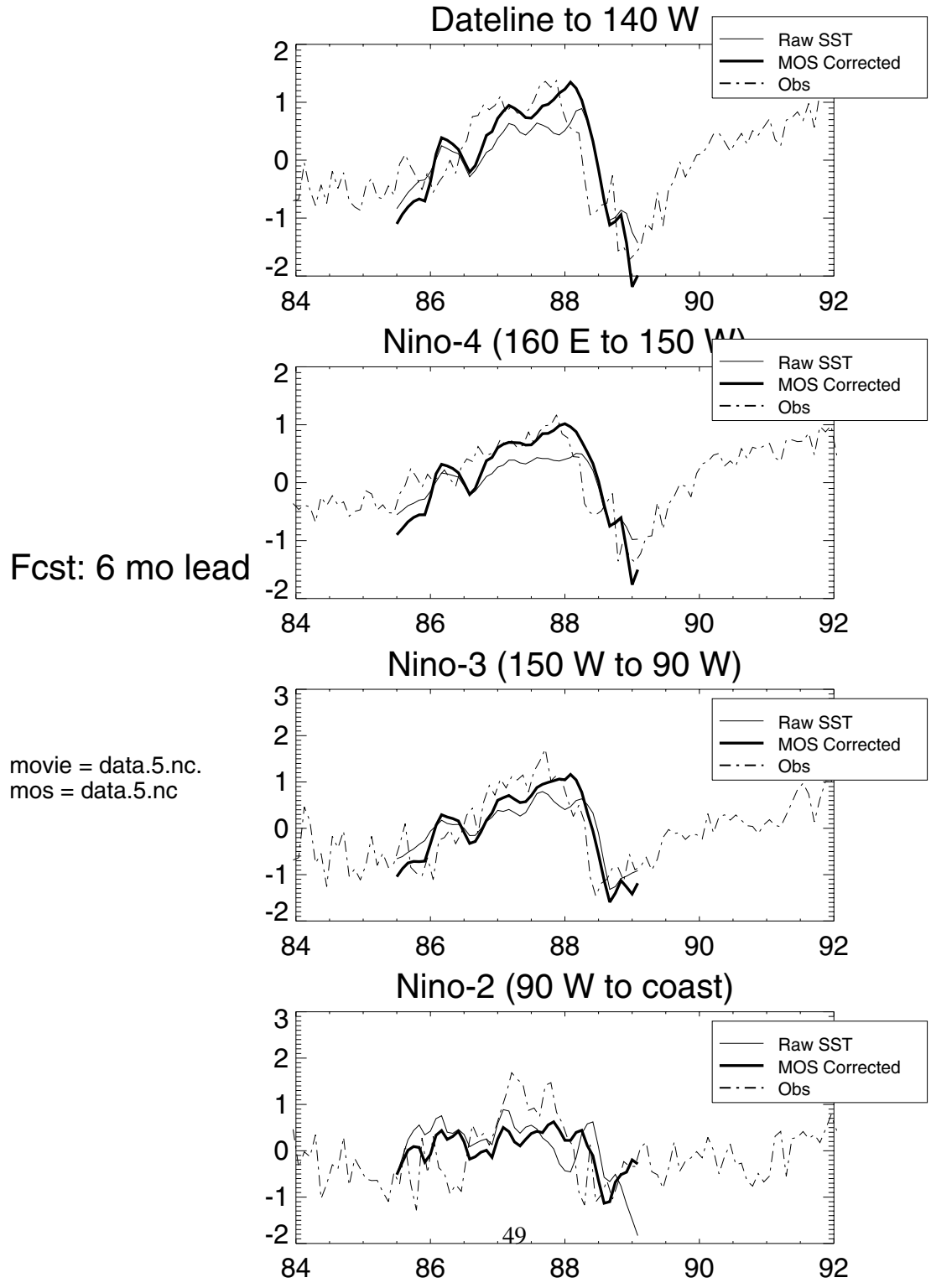

Figure 27: Forecast at 6 month lead time.

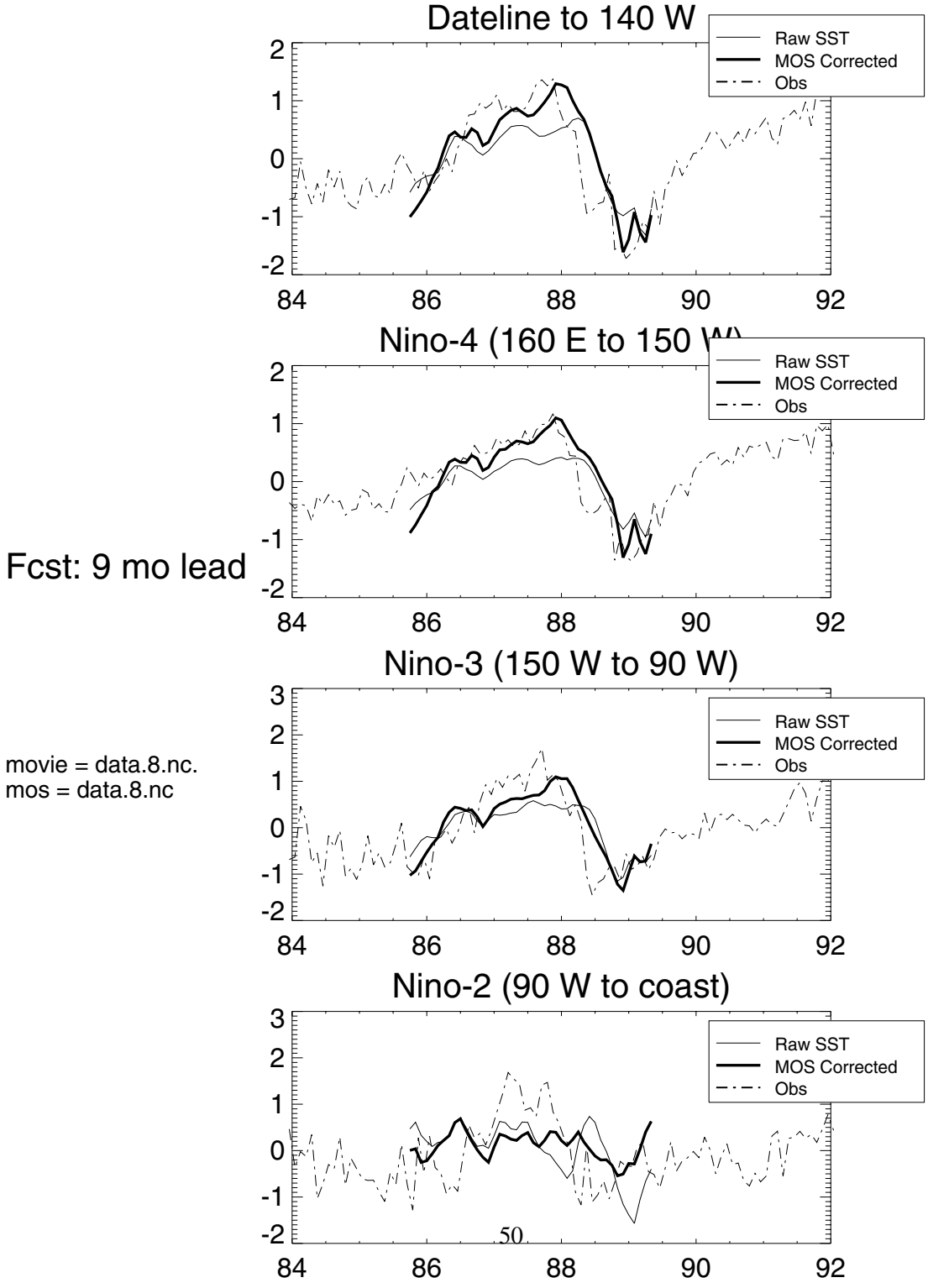

Figure 28: Forecast at 9 month lead time.

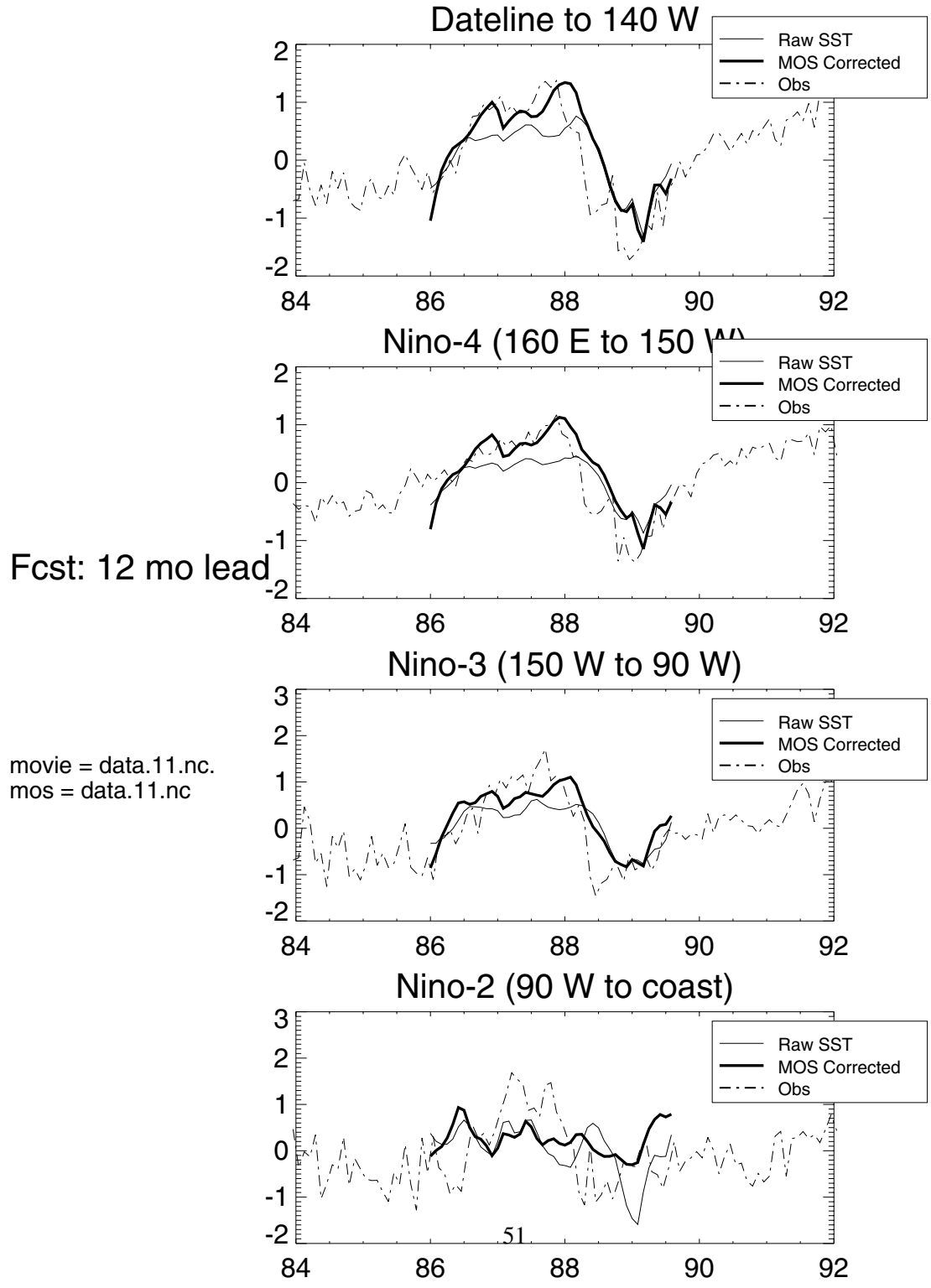

Figure 29: Forecast at 12 month lead time.

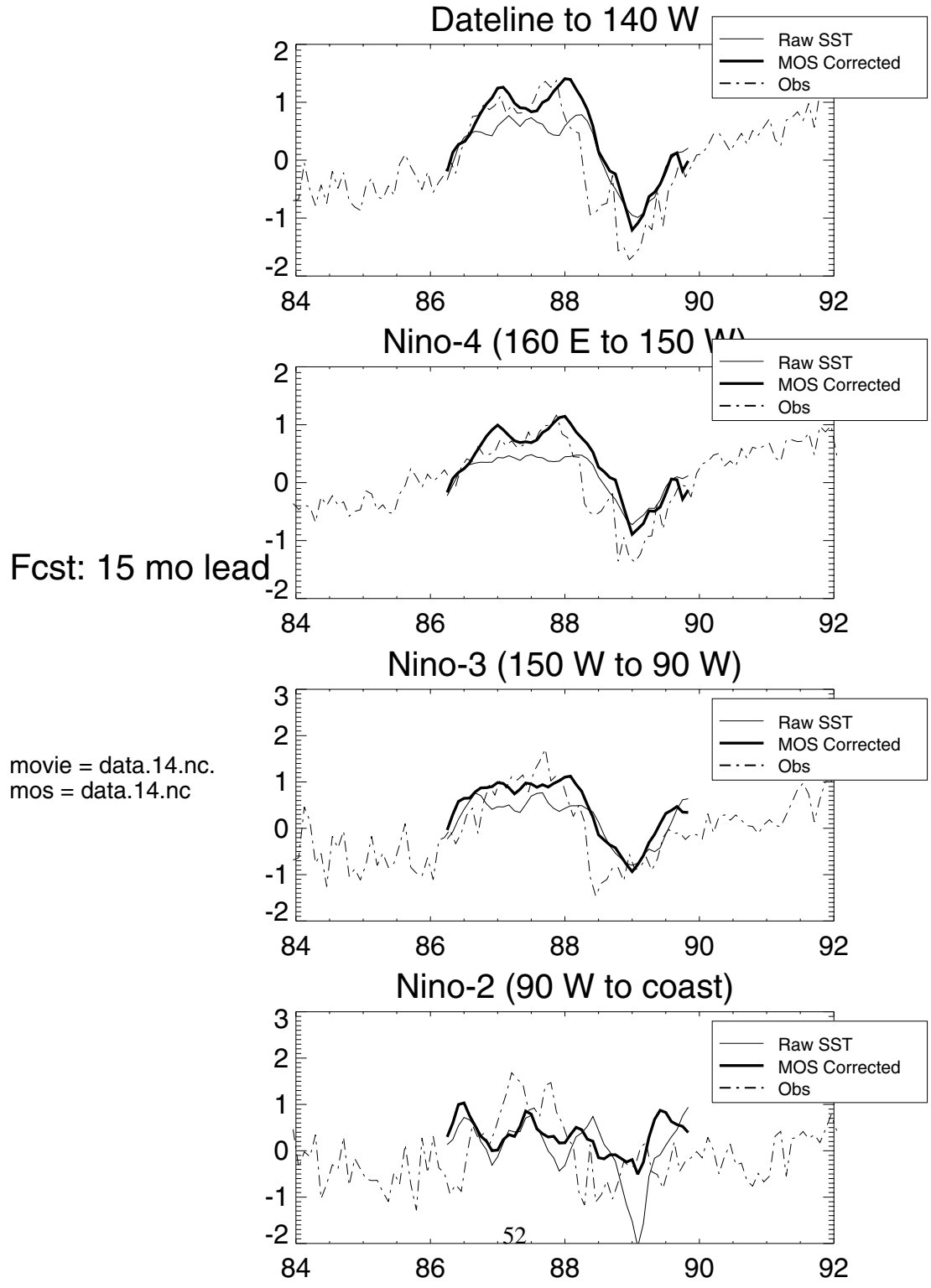

Figure 30: Forecast at 15 month lead time.

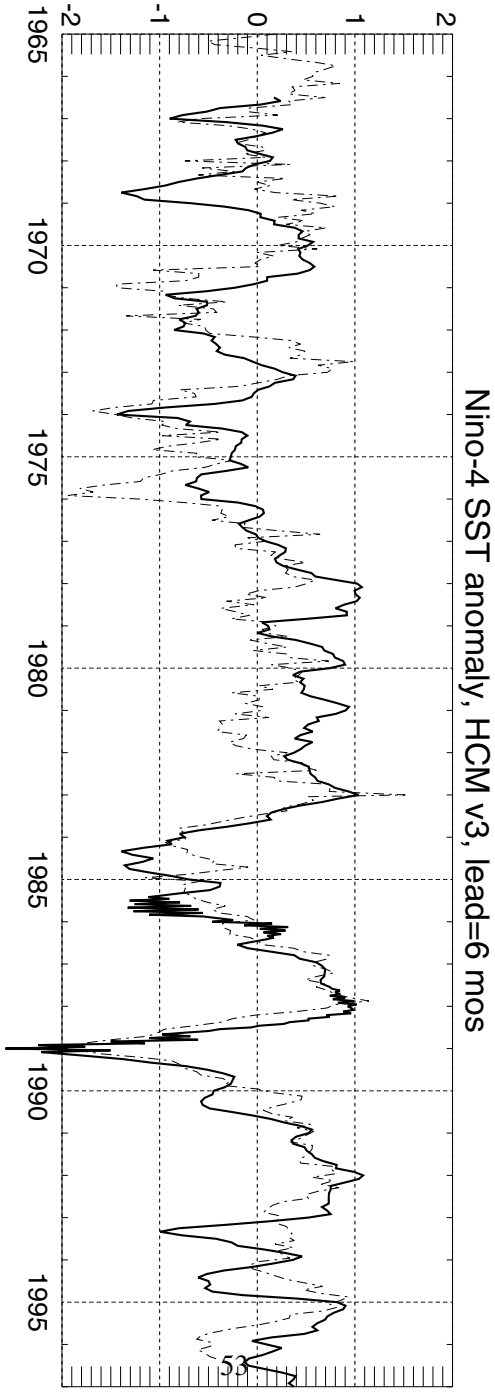

Figure 31: Forecast at 6 month lead time. Solid is model; dashed is observations.

A. LAPACK-based EOF program

```
      subroutine deof( data, nx, nt, nmodes, icovcor,
     &          eigenvalues, eigenvectors, princomp, variance, cumvariance,
     &          iminnxnt, imaxnxnt, spacked, evals, evecs, rlawork,
     &          ilawork, ifail, tentpcs, sqrootweights )

c       ----------------------------------------------------------------
c       This routine generates the Empirical Orthogonal Functions (EOFs)
c       for a given time-space data set.  (Note that although the routine
c       is set up as if the data is exactly two dimensional (X,T), that
c       you can compute the EOFs of 3-D (X,Y,T) fields simply by concat-
c       enating the data along the X,Y axes into one big 2-D array and
c       then computing the EOFs of that 2-D array).  The "typical"
c       normalization is applied, i.e., the magnitude of the eigenvectors
c       is scaled to be equal to 1.
c
c       David W. Pierce
c       dpierce@ucsd.edu
c       Scripps Institution of Oceanography
c       Climate Research Division, 0224
c       Jan 29, 1996
c
c       Inputs:
c
c               data(nx,nt): data to compute the EOFs of.  THIS MUST
c                       BE ANOMALIES.
c
c               nx, nt: number of space (nx) and time (nt) points in the
c                       input data array.
c
c               nmodes: number of modes (EOFs) to compute.
c
c               icovcor: if =0, then compute using covariance matrix;
c                        if =1, then compute using correlation matrix.
c                       See, for example, discussion in Daniel S. Wilks
c                       1995, "Statistical Methods in the Atmospheric
c                       Sciences", p. 383 for a discussion.
c
c               iminnxnt: the SMALLER of 'nx' and 'nt'.
c
c               imaxnxnt: the LARGER of 'nx' and 'nt'.
c
c               spacked(iminnxnt*(iminnxnt+1)/2): workspace.  This is used to
c                       store the packed covariance or correlation matrix.
c
c               evals(iminnxnt): workspace.  This is used to store the
c                       complete series of eigenvalues.  I suppose you
c                       could access this if you wanted to, remembering
c                       that a) it's more modes than you asked for; b)
c                       they are in ASCENDING order rather than descending.
c
c               evecs(iminnxnt,nmodes): workspace.  This is used to store
c                       the (possibly switched) eigenvectors, which are
```

```
c                       in ascending order.
c
c               rlawork(8*iminnxnt): Real LAPACK workspace.
c
c               ilawork(5*iminnxnt): Integer LAPACK workspace.
c
c               ifail(iminnxnt): Integer LAPACK workspace.  This is used
c                       to indicate which eigenvectors didn't converge,
c                       if that happened.
c
c               tentpcs(imaxnxnt,nmodes): Real workspace.  This is
c                       used to store the 'tentative' principal components.
c
c               sqrootweights(nx) : the SQUARE ROOT of the areal weighting to
c                       use.  Just set this to all 1.0's if you don't care
c                       about areal weighting.  I think you can set places
c                       which should be ignored to zero to have them not
c                       participate in the calculation, thus implementing a
c                       crude form of data masking.
c
c       Outputs:
c
c               eigenvalues(nmodes): the computed eigenvalues of
c                       the data, the largest returned first.
c
c               eigenvectors(nx,nmodes): the computed eigenvectors
c                       of the data, the most important returned first.
c
c               princomp(nt,nmodes): the principal components of
c                       the data.
c
c               variance(nmodes): the percent of variance explained
c                       by each mode.
c
c               cumvariance(nmodes): the cumulative percent of
c                       variance explained by each mode.  This is
c                       a bit redundant -- it's just the sum
c                       of "variance".
c
c       Method:
c
c       EOFs are simply the eigenvectors of the covariance (or correlation)
c       matrix.  So form the proper matrix from the data and pass it to
c       a LAPACK routine which calculates eigenvectors/eigenvalues.  Then
c       calculate the principal components explicitly using the fact that
c       the principal component for mode M at time T is just the projection
c       of the data at time T onto eigenvector M.
c
c       There is a slight complication for efficiency's sake.  That is,
c       we often have nx >> nt, i.e., there are many more spatial than
c       temporal points.  The traditional correlation/covariance matrix
c       is size (nx,nx), which can be huge.  Think, for example, of a 3-D
c       field of size (100,100,120) which has been concatenated into a
c       2-D field of size (10000,120).  The traditional covariance/correlation
c       matrix in this case is size (10000,10000)!  That's way too big to
```

```
c       easily work with.  So, following the discussion in Preisendorfer
c       (1988, "Principal Component Analysis in Meteorology and Oceanography",
c       p. 64) we work, in such cases, in the "dual" of the space.  All
c       that means is that we logically switch X and T; the
c       covariance/correlation matrix for the example given above is then
c       size (120,120), which is much easier and more efficient to work
c       with.  If you do this kind of switch, the basic idea is that
c       the eigenvectors of the switched matrix are the principal components
c       of the original matrix, and the principal components of the switched
c       matrix are the eigenvectors of the original matrix.  Which is to
c       say, if you switch T and X to begin with, the final result is that
c       the T dependence is in the X part and the X dependence is in the
c       T part.  There is also a normalization which has to be applied--
c       see Preisendorfer for details.
c       ------------------------------------------------------------------

        implicit none

        integer covariance, correlation
        parameter( covariance=0, correlation=1 )  ! possible values of 'icovcor'

c       -----------------
c       Passed parameters
c       -----------------
        integer nx, nt, nmodes, icovcor, iminnxnt, imaxnxnt, ifail(iminnxnt)
        double precision data(nx,nt), eigenvalues(nmodes),
     &          eigenvectors(nx,nmodes), princomp(nt,nmodes),
     &          variance(nmodes), cumvariance(nmodes),
     &          spacked(iminnxnt*(iminnxnt+1)/2), evals(iminnxnt),
     &          evecs(iminnxnt,nmodes), rlawork(8*iminnxnt),
     &          tentpcs(imaxnxnt,nmodes), sqrootweights(nx)
        integer ilawork(5*iminnxnt)

c       ---------------
c       Local variables
c       ---------------
        logical         doswitched
        integer         orderofs, i, j, jascending, jdescending
        double precision sum, fact, totvar
        character*1     jobz, range, uplo       ! for LAPACK routine
        integer         m, n, il, iu, ldz, info ! for LAPACK routine
        double precision vl, vu, abstol         ! for LAPACK routine
        integer         lwork, mode

#ifdef VERBOSE
        print *, 'entering deof with nx, nt, nmodes=', nx, nt, nmodes
#endif
c       ---------------------------------------------
c       Weight the data by the square root of the area
c       ---------------------------------------------
        do j=1, nt
        do i=1, nx
                data(i,j) = data(i,j) * sqrootweights(i)
        enddo
        enddo
```

```
c       ---------------------------------------------------------
c       Figure out whether we should do the 'regular' EOF or the
c       one with switched X and T axes.  The correlation matrix
c       for the regular method is size (nx,nx) and for the
c       switched method it is size (nt,nt); choose based on which
c       of these is smaller.
c       ---------------------------------------------------------
#ifdef VERBOSE
        print *, 'figuring switched or not'
#endif
        doswitched = (nx .gt. nt)
        if( doswitched ) then
                orderofs = nt
                print *, 'deof: Working in switched mode'
        else
                orderofs = nx
                print *, 'deof: Working in unswitched mode'
        endif
        if( orderofs .gt. iminnxnt ) then
                write(0,*) 'Error!  EOF routine must be supplied '
                write(0,*) 'with enough workspace; passed parameter'
                write(0,*) 'iminnxnt must be at least ', orderofs
                write(0,*) 'Passed value was iminnxnt=',iminnxnt
                stop 'eof.F near line 145'
        endif
        if( nmodes .gt. orderofs ) then
                write(0,*) 'Error! EOF routine called requesting more'
                write(0,*) 'modes than exist!  Request=',nmodes,' exist=',
     &                  orderofs
                stop 'eof.F near line 170'
        endif

c       -------------------------------------------------
c       Form the covariance or correlation matrix, put it
c       into 's'.  Note that 's' is always symmetric --
c       the correlation between X and Y is the same as
c       between Y and X -- so use packed storage for 's'.
c       The packed storage scheme we use is the same as
c       LAPACK uses so we can pass 's' directly to the
c       solver routine: the matrix is lower triangular,
c       and s(i,j) = spacked(i+(j-1)*(2*n-j)/2).
c       -------------------------------------------------
        call deofcovcor( data, nx, nt, icovcor, covariance,
     &          correlation, spacked, doswitched, iminnxnt )

c       -----------------------------------------------------
c       Now call the LAPACK solver to get the eigenvalues
c       and eigenvectors.  The eigenvalues express the
c       amount of variance explained by the various modes,
c       so choose to return those 'nmodes' modes which
c       explain the most variance.
c       Dims of arrays:
c               evals  (n)        ! The calculated eigenvalues
c               evecs  (n,nmodes) ! The calculated eigenvectors
```

```
c               rlawork(8*n)
c               ilawork(5*n)
c               ifail  (n)
c       Remember that the calculated eigenvectors may not be
c       the ones we really want if we are doing switched
c       X and T axes.  However the eigen*values* are the same
c       either way, according to Preisendorfer.
c       *NOTE* that the LAPACK routine returns the eigenvalues
c       (and corresponding eigenvectors) in ASCENDING order,
c       but this routine returns them in DESCENDING order;
c       this will be switched in the final assembly phase,
c       below.
c       ----------------------------------------------------
        jobz  = 'V'     ! Both eigenvalues and eigenvectors
        range = 'I'     ! Specify range of eigenvalues to get.
        uplo  = 'L'     ! 'spacked' has lower triangular part of s
        n     = orderofs
        il    = n - nmodes + 1  ! Smallest eigenvalue to get
        iu    = n               ! Largest eigenvalue to get
        abstol= 0.0             ! See LAPACK documentation
        ldz   = n

        lwork = 8*iminnxnt

        do i=1, nmodes
                evals(i) = 0.0
        enddo
        do j=1, nmodes
        do i=1, iminnxnt
                evecs(i,j) = 0.0
        enddo
        enddo

        call dspevx( jobz, range, uplo, n, spacked,
     &          vl, vu, il, iu, abstol, m, evals, evecs, ldz,
     &          rlawork, ilawork, ifail, info )

c       ------------------------------
c       Check for LAPACK routine error
c       ------------------------------
        if( info .ne. 0 ) then
                if( info .lt. 0 ) then
                        write(0,*) 'LAPACK error: argument ',
     &                          -info, ' had illegal value'
                        stop 'eof.F near line 167'
                else
                        write(0,*) 'LAPACK error: ', info,
     &                          'eigenvectors failed to converge!'
                        write(0,*) 'Consult the LAPACK docs!'
                        stop 'eof.F near line 172'
                endif
        endif

c       ------------------------------------------------
c       Make sure that no eigenvalues <= zero.  Besides
```

```
c       being mathematically forbidden, this would cause
c       a divide by zero or negative sqrt error later on.
c       ------------------------------------------------
        do i=1, nmodes
                if( evals(i) .le. 0.0 ) then
                        write(0,*) 'Error! LAPACK routine returned'
                        write(0,*) 'eigenvalue <= 0!! ', i, evals(i)
                        do j=1, nmodes
                                print *, j, evals(j)
                        enddo
                        write(0,*) 'Note: This often means you are asking'
                        write(0,*) 'for more modes than the data supports.'
                        write(0,*) 'Try reducing the number of requested'
                        write(0,*) 'modes.'
                        stop 'eof.F near line 197'
                endif
        enddo

c       ------------------------------------------------
c       Compute the tentative principal components; they
c       are 'tentative' because they might be the PCs
c       of the switched data.  Put them into 'tentpcs',
c       which is of size (nt,nmodes) if we are doing
c       regular EOFs and (nx,nmodes) if we are doing
c       switched EOFs.  These PCs come out in order
c       corresponding to the order of 'evecs', which is
c       in ASCENDING order.
c       ------------------------------------------------
        call deofpcs( data, nx, nt, nmodes, iminnxnt,
     &          imaxnxnt, evecs, doswitched, tentpcs )

c       ------------------------------------------------
c       Now we have all the pieces to assemble our final
c       result.  How we actually assemble them depends on
c       whether we are doing switched or unswitched EOFs
c       (except for the eigenVALUES, which are the same
c       either way).
c       ------------------------------------------------
        if( doswitched ) then
c               ----------------------------------------
c               In this case we must switch the principal
c               components and the eigenvectors, applying
c               the proper normalization.
c               First get the unswitched eigenvectors,
c               which are the switched (tentative) principal
c               components divided by the square root of the
c               appropriate eigenvalue.  Recall that the
c               LAPACK values are in ASCENDING order while
c               we want them in DESCENDING order; do the
c               switch in this loop.
c               ----------------------------------------
                do jascending=1, nmodes
                        jdescending = nmodes - jascending + 1
                        fact        = 1.0/sqrt(evals(jascending))
                        do i=1, nx
```

```
                                eigenvectors(i,jdescending) =
     &                                  tentpcs(i,jascending)*fact
                        enddo
                enddo

c               ------------------------------------------------
c               Next get unswitched principal components, which
c               are the switched eigenvectors multiplied by
c               the appropriate eigenvalues.
c               ------------------------------------------------
                do jascending=1, nmodes
                        jdescending = nmodes - jascending + 1
                        fact        = sqrt(evals(jascending))
                        do i=1, nt
                                princomp(i,jdescending) =
     &                                  evecs(i,jascending)*fact
                        enddo
                enddo
        else
c               ------------------------------------------------
c               This is the unswitched case, and so it is easier.
c               All we have to do is return things in DESCENDING
c               order despite the fact that LAPACK returns them
c               in ASCENDING order.
c               Do the eigenvectors first...
c               ------------------------------------------------
                do jascending=1, nmodes
                        jdescending = nmodes - jascending + 1
                        do i=1, nx
                                eigenvectors(i,jdescending) =
     &                                  evecs(i,jascending)
                        enddo
                enddo

c               --------------------------------
c               ...then the principal components
c               --------------------------------
                do jascending=1, nmodes
                        jdescending = nmodes - jascending + 1
                        do i=1, nt
                                princomp(i,jdescending) =
     &                                  tentpcs(i,jascending)
                        enddo
                enddo


        endif

c       ----------------------------------------
c       Do the second half of the areal weighting...
c       ----------------------------------------
        do mode=1, nmodes
        do i=1, nx
                if( sqrootweights(i) .eq. 0.0 ) then
                        eigenvectors(i,mode) = 0.0
```

```
                else
                        eigenvectors(i,mode) =
     &                          eigenvectors(i,mode) / sqrootweights(i)
                endif
           enddo
           enddo

c          ------------------------------------------------
c          Scale the eigenvectors to have a magnitude of 1;
c          scale the corresponding principal components to
c          reproduce the original data.
c          ------------------------------------------------
           do mode=1, nmodes
c                  ----------------------------
c                  Get the normalization factor
c                  ----------------------------
                   sum = 0.0
                   do i=1, nx
                           sum = sum + eigenvectors(i,mode)*eigenvectors(i,mode)
                   enddo
                   fact = sqrt(sum)
c                  -------------------------
c                  Normalize the eigenvectors
c                  -------------------------
                   do i=1, nx
                           eigenvectors(i,mode) = eigenvectors(i,mode)/fact
                   enddo
c                  ---------------------------------
c                  Normalize the principal components
c                  ---------------------------------
                   do i=1, nt
                           princomp(i,mode) = princomp(i,mode)*fact
                   enddo
           enddo

c          ------------------------------------------------
c          Copy over just the requested number of
c          eigenvalues, and calculate the cumulative percent
c          variance explained.  Start by getting the total
c          variance in the field, so we can normalize by
c          that.
c          ------------------------------------------------
           call deoftotvar( data, nx, nt, totvar, doswitched,
     &          icovcor, covariance, correlation )
           sum = 0.0
           do jascending=nmodes, 1, -1
                   jdescending = nmodes - jascending + 1
                   eigenvalues(jdescending) = evals(jascending)
                   variance(jdescending)    =
     &                   eigenvalues(jdescending)/totvar*100.0
                   sum = sum + variance(jdescending)
                   cumvariance(jdescending) = sum
           enddo

           return
```

```
        end

c=======================================================================

        subroutine deofcovcor( data, nx, nt, icovcor, covariance,
     &          correlation, spacked, doswitched, iminnxnt )

c       ------------------------------------------------
c       Form the covariance or correlation matrix, put it
c       into 's'.  Note that 's' is always symmetric --
c       the correlation between X and Y is the same as
c       between Y and X -- so use packed storage for 's'.
c       The packed storage scheme we use is the same as
c       LAPACK uses so we can pass 's' directly to the
c       solver routine: the matrix is lower triangular,
c       and s(i,j) = spacked(i+(j-1)*(1*n-j)/2).
c
c       Inputs:
c               data(nx,nt): The basic data array.  THIS MUST
c                       BE ANOMALIES.
c
c               nx, nt: size of data array
c                       [INTEGER]
c
c               icovcor: if .eq. covariance, then calculate
c                       the covariance array;
c                        if .eq. correlation, then calculate
c                       the correlation array.
c                       [INTEGER]
c
c               covariance, correlation: integer values to
c                       indicate each of these options.
c                       [INTEGER]
c
c               doswitched: if .TRUE., then calculate the
c                       'switched' array (which is of size
c                       (nt,nt)); if .FALSE., then calculate
c                       the normal array of size (nx,nx).
c                       [LOGICAL]
c
c               iminnxnt: min(nt,nx).  Used to dimension
c                       'spacked'.
c
c       Outputs:
c
c               spacked(iminnxnt*(iminnxnt+1)/2): the covariance
c                       or correlation array.  This is in packed
c                       form corresponding to LAPACK's lower
c                       triangular form.
c
c       David Pierce
c       Scripps Institution of Oceanography
c       Climate Research Division
c       dpierce@ucsd.edu
c       Jan 29, 1996
```

```
c       ------------------------------------------------

        implicit none

c       -----------------
c       Passed parameters
c       -----------------
        integer nx, nt, icovcor, covariance, correlation, iminnxnt
        double precision data(nx,nt), spacked(iminnxnt*(iminnxnt+1)/2)
        logical doswitched

c       ---------------
c       Local variables
c       ---------------
        integer i, j, k
        double precision sum, sum2, sum3, fact

        if( doswitched ) then
                do j=1, nt
                do i=j, nt
                        sum  = 0.0
                        sum2 = 0.0
                        sum3 = 0.0
                        do k=1, nx
                                sum  = sum  + data(k,i)*data(k,j)
                                sum2 = sum2 + data(k,i)*data(k,i)
                                sum3 = sum3 + data(k,j)*data(k,j)
                        enddo
                        if( icovcor .eq. covariance ) then
                                fact = 1.0/float(nx-1)
                        else
                                fact = 1.0/(sqrt(sum2)*sqrt(sum3))
                        endif
                        spacked(i+(j-1)*(2*nt-j)/2) = sum*fact
                enddo
                enddo
        else
                do j=1, nx
                do i=j, nx
                        sum  = 0.0
                        sum2 = 0.0
                        sum3 = 0.0
                        do k=1, nt
                                sum  = sum  + data(j,k)*data(i,k)
                                sum2 = sum2 + data(i,k)*data(i,k)
                                sum3 = sum3 + data(j,k)*data(j,k)
                        enddo
                        if( icovcor .eq. covariance ) then
                                fact = 1.0/float(nt-1)
                        else
                                fact = 1.0/(sqrt(sum2)*sqrt(sum3))
                        endif
                        spacked(i+(j-1)*(2*nx-j)/2) = sum*fact
                enddo
                enddo
```

```
        endif

        return
        end

c=======================================================================

        subroutine deofpcs( data, nx, nt, nmodes, iminnxnt,
     &          imaxnxnt, evecs, doswitched, tentpcs )

c       ------------------------------------------------
c       Compute the tentative principal components; they
c       are 'tentative' because they might be the PCs
c       of the switched data.  Put them into 'tentpcs',
c       which is of size (nt,nmodes) if we are doing
c       regular EOFs and (nx,nmodes) if we are doing
c       switched EOFs.  These PCs come out in order
c       corresponding to the order of 'evecs', which is
c       in ASCENDING order.
c
c       Inputs:
c
c               data(nx,nt): The input data.  THESE MUST
c                       BE ANOMALIES.
c
c               nmodes: # of modes to calculate.
c
c               iminnxnt: min(nx,nt)
c
c               evecs(iminnxnt,nmodes): the eigenvectors
c                       (which might be switched).
c
c               doswitched: if .TRUE., then we are doing
c                       switched (space,time) calculation;
c                       otherwise, regular (time,space)
c                       calculation.
c
c       Outputs:
c
c               tentpcs(imaxnxnt,nmodes): the tentative
c                       (possibly switched) principal
c                       components.
c
c       David W. Pierce
c       Scripps Institution of Oceanography
c       Climate Research Division
c       dpierce@ucsd.edu
c       Jan 29, 1996
c       ------------------------------------------------

        implicit none

c       -----------------
c       Passed parameters
c       -----------------
```

```
      integer nx, nt, nmodes, iminnxnt, imaxnxnt
      double precision data(nx,nt), evecs(iminnxnt,nmodes),
     &         tentpcs(imaxnxnt,nmodes)
      logical doswitched

c     ---------------
c     Local variables
c     ---------------
      integer i, j, k
      double precision        sum

      if( doswitched ) then
              do j=1, nmodes
              do i=1, nx
                      sum = 0.0
                      do k=1, nt
                              sum = sum + data(i,k)*evecs(k,j)
                      enddo
                      tentpcs(i,j) = sum
              enddo
              enddo
      else
              do j=1, nt
              do i=1, nmodes
                      sum = 0.0
                      do k=1, nx
                              sum = sum + data(k,j)*evecs(k,i)
                      enddo
                      tentpcs(j,i) = sum
              enddo
              enddo
      endif

      return
      end
```

*References

Barnett, T. P., M. Latif, N. Graham, M. Flugel, S. Pazan, and W. White, 1993: ENSO and ENSO-related Predictability. Part I: Predictions of Equatorial Pacific Sea Surface Temperature with a Hybrid Coupled Ocean-Atmosphere Model. *J. Climate*, 6, 1545–1566.

Barnett, T. P., M. Latif, E. Kirk, and E. Roeckner, 1991: On ENSO Physics . *J. Climate*, 4, 487–515.

Preisendorfer, R. W., 1988: Principal component analysis in meteorology and oceanography . Elsevier, 425 pp.

Schopf, P. S., 1983: On Equatorial Waves and El Niño. II: Effect of Air-Sea Thermal Coupling. *J. Phys. Oceanogr.*, 13, 1878–1893.