MET O 11 TECHNICAL NOTE NO. 149

A MESOSCALE ANALYSIS SCHEME EXPLICITLY ENCOURAGING THE INCORPORATION OF SHARP DISCONTINUITIES - PRELIMINARY REPORT DESCRIBING THE FUNCTIONS OF COMPUTER PROGRAMS

by

M.R. MACFADYEN

Met O 11
Meteorological Office
London Road
Bracknell
Berkshire

August 1981

FH2A

JC

A MESO-SCALE ANALYSIS SCHEME EXPLICITLY ENCOURAGING THE INCORPORATION OF SHARP DISCONTINUITIES - PRELIMINARY REPORT DESCRIBING THE FUNCTIONS OF COMPUTER PROGRAMS.

BY M R MACFADYEN (MET O 11)

One of the fundamental problems of meso-scale forecasting is that data networks are typically much sparser than the grid points of the model, so that many of the features accurately resolvable by the model cannot easily be analysed from real data.

Most numerical analysis schemes involve some sort of smoothing in interpolating between observations; some of the most important meso-scale meteorological phenomena, however, involve rather sharp discontinuities. The capacity to produce analyses incorporating such sharp discontinuities might, therefore, be a useful step towards effectively initialising a meso-scale model.

This note describes a programme which has been written for the purposes of such analysis.

GENERAL STRUCTURE:

Broadly, the programme works in three steps. First, using synoptic data, it produces a large number of estimates for the positions of dividing lines between air masses. These might be for example trough lines, lines separating geographically close stations of widely differing dew points, or positions just downstream of stations whose barometers have recently "kicked". Each of these estimates is expressed as a position, with an orientation, and estimates of the accuracy of each of these.

The second phase of activity is to take these estimated positions and knit them together into long lines, which will be termed 'fronts' hereinafter, though they do not necessarily accord closely with the classical notion of a front. They are sets of points such that each set begins and ends either on a wall or on another front. There are, however, no guarantees at this stage that they do not cross and recross each other or divide off very small areas.

The third phase consists of a topological rationalisation of the output of the second. In order to avoid geometrical problems we transfer to a grid notation, and label the points along the fronts with sufficient density that the areas in between them are separated (in the sense that no pair of orthogonally adjacent points in different areas is unlabelled). Having labelled the 'fronts' in this way it is a relatively simple matter to label the spaces between them, after which a number of algorithms are employed to simplify the topology. For example very small areas are amalgamated with some large area adjacent to them.

The whole process takes about 1½ seconds of 360/195 time for 40 data points with a 20 x 30 point grid.

No programmes currently exist to use the output of this scheme for analysis. Possible uses for it include the following:

- In a boundary layer scheme which operates by setting flags to characterise one of a dozen different boundary layer types (eg fog/low stratus; shallow daytime convective etc) and operates a different parametric scheme for each of them, the programme may be useful to define boundaries between the types.
- The present programme could easily be adjusted so that it retains some grid points as being explicity part of fronts. At these points a deep convection scheme appropriate to, for example, an active cold front might be employed. This would depend, of course, on the type of deep convection parameterisation being employed.

The following text divides into four sections. Each of the three phases of the programme is described in general terms, followed by a more detailed description of the working of its subroutines, and the account is concluded with a discussion of the omissions and weaknesses of the programme as at present constituted, with some ideas as to how it might be improved.

<u>TERMINOLOGY</u>:

The following terms describe the main variables to be used in characterising the 'fronts'.

Segment: One of the elements from which fronts are to be constructed, stored as a set of five numbers:

$\left.\begin{matrix} x \\ y \end{matrix}\right\}$ position

θ angle - fixed so that $0 \leqslant \theta < \pi$

σ_x tolerance on position

σ_θ tolerance on angle

Front: An ordered set of points generated by the programmes to be described. Each point is characterised by three variables :

$\left.\begin{matrix} x \\ y \end{matrix}\right\}$ position

σ_x tolerance on position

Tolerances are typically calculated by imprecise algorithms, but are intended to be roughly one standard deviation in the uncertainty of the variable in question.

PHASE I

Routines to extract estimates of frontal position from meteorological data are at present relatively poorly developed, and are designed to run on a limited data set consisting of pressure, wind and pressure tendency observations for one hour, together with wind observations for the previous hour.

The following subroutines are used to detect troughs:

WVEER

This subroutine compares the present wind direction with that for the previous hour at each station. If the wind has veered by more than a specified (in subroutine call) amount, then a segment of front is generated at a position ½-hour downstream from the station at the mean wind speed for the two hours, with its angle normal to the mean wind. The error on the position of the segment is taken as the mean wind speed x 1 hour. The error on its angle is taken as the amount by which the wind has veered.

BARKIK

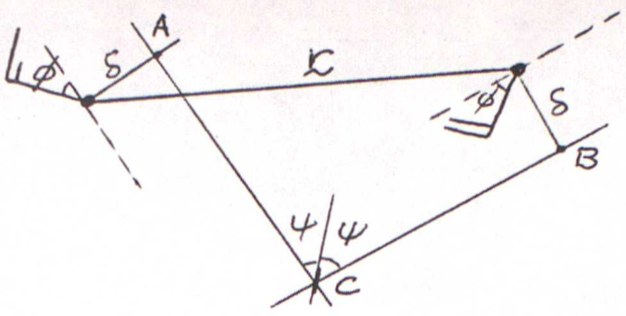
This subroutine identifies stations reporting pressure tendency code 5 and produces a segment ½-hour downstream of them at the current wind speed, with its angle normal to the wind.

The error on position is taken as ½ hour x the wind speed. The error on angle is fixed at $\pi/2$

TROUGH

This subroutine seeks pairs of stations whose relative wind direction implies a trough, and locates it.

The construction proceeds as follows:



- Only those pairs of stations the component of whose wind direction along the vector joining them has the same sign are considered.
- Wind directions are corrected by an angle ϕ, currently set at 0.25 radians, so as to approximate the direction of the isobars better.
- The distance δ is calculated from the pressure difference between the stations as an adjustment to reach the points A and B at which the pressures should be equal.
- The 'isobars' are then extrapolated to C, where they cross, and the angle between them is bisected to give the estimated direction of the frontal segment.
- The error on the angle of the trough is taken as $\frac{1}{2}(\pi - 2\psi)$. The error on its position as $0.1 \times (r_x + r_y)/\sin 2\psi$, where r_x and r_y are the components of $\underset{\sim}{r}$.

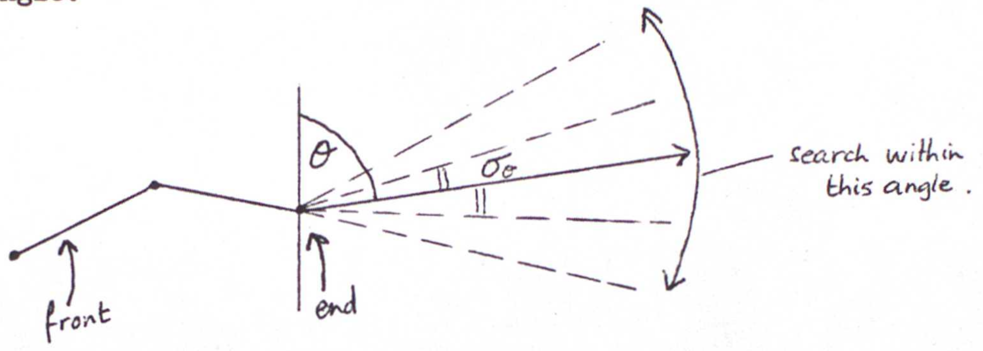
PHASE II.

The segments estimating trough positions are 'knitted' into fronts using a directionalised cluster analysis which it is the purpose of this section to describe. The basic tool used is a subroutine called SEGFIT which calculates the goodness of fit between a pair of segments. This is a number between 0 and 1 which is large for pairs of segments close together, and whose directions point along the line joining them. The subroutine also calculates the coordinates for a new segment representing the best fit to the old pair. It is described in more detail later.

The cluster analysis begins by calculating the goodness of fit between every pair of segments. The group of segments which fits together best is then isolated. The number of members of this group is a control variable which could be altered within the programme but is fixed in the current version.

Having isolated the best group we start building a front with it. At each stage during its construction the front will consist of an ordered string of points, both ends of which are stored separately.

Further segments to be used for extending the front are sought by measuring their goodness of fit with an end of the front. All segments whose goodness of fit exceed a threshold value are concatenated to form a single segment, which becomes the next point on the front. The search is restricted to segments within $2\sigma_\theta$ of the advertised angle.



Further points are sought until an end is reached for which no points within the required tolerance can be found, or until the array allocated for storing the front is full (24 points). The two ends are extended in this way in turn, and all the segments which have been used are flagged to prevent their being used again.

The programme then returns to seeking another group of segments with which to start building a front, and continues until no groups fit sufficiently well, or until ten fronts have been generated.

The next stage is to extrapolate the ends of the fronts so that they meet either walls or each other. This is necessary in order that they should separate the field into distinct areas. Two subroutines are employed for this purpose. The first finds the nearest segment of front to the end, and the second, for cases when the first finds no intersections, projects the end onto a wall.

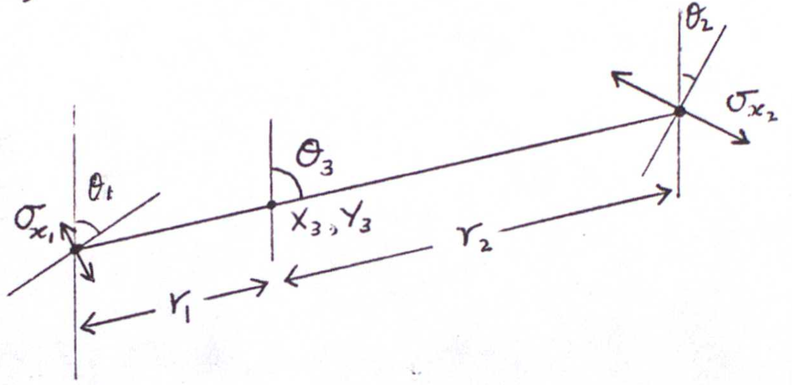
This process completes the initial generation of fronts, and also the main programme, so we pause here to describe some of the details of the subroutines which have been used.

SEGFIT

This subroutine finds a line segment which fits the two segments provided, and calculates how well they fit it.

The two segments each have a position and an angle, so we have four variables to attempt to fit.

The positions are dealt with by taking a line between the two segments and dividing it in proportion to the ratio of variances on position of the segments. This gives the point X_3, Y_3 as shown



We now have three estimates for the angle of the new segment; θ_1 and θ_2 as provided, and θ_3 which is the direction of the line joining the two segments. The formula used is:

$$\phi_{OUT} = \left(\frac{2\theta_3}{\sigma_3} + \frac{\theta_1}{\sigma_{\theta_1}} + \frac{\theta_2}{\sigma_{\theta_2}}\right) \Big/ \left(\frac{2}{\sigma_3} + \frac{1}{\sigma_{\theta_1}} + \frac{1}{\sigma_{\theta_2}}\right)$$

where $$\sigma_3 = \left(\frac{\sigma_{x_1}}{r_1}\right) = \left(\frac{\sigma_{x_2}}{r_2}\right)$$

Supposing the σ values to represent one standard deviation of a normally distributed variable, we then obtain a goodness of fit

$$P = E\left(\frac{\phi_{OUT} - \theta_1}{\sigma_1}\right) \times E\left(\frac{\phi_{OUT} - \theta_2}{\sigma_2}\right) \times \left\{E\left(\frac{\phi_{OUT} - \theta_3}{\sigma_3}\right)\right\}^2$$

where E is the error function

$$E(x) = \frac{1}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt = ERFC(x) \text{ in Fortran.}$$

This number P is modified so that remote points fit with lower probability, using a characteristic distance which is also one of the calling variables.

CONCAT

This subroutine combines a number of segments into a single one using subroutine SEGFIT.

It operates by searching through the segments to find the pair which fit with the highest probability, and then combining them, thus reducing the number of segments by one, and repeating the process until only one is left.

This apparently rather inefficient procedure (compared with simply combining them in pairs irrespective of goodness of fit) is necessary in order to avoid losing information in the case where some of the points fit together much better than others.

NODES

This subroutine finds the intersection of a line projecting from a specified point at a specified angle with a front.

The process is to calculate the angle subtended by each point on the front at the end, and then to compare these angles successively with the angle at which the end projects. Whenever this comparison shows a change of sign, a crossing point is calculated. The closest of these crossing points is selected as output.

WALLPT

This is a simple-minded subroutine to find the intersection of a line projecting from a specified point at a specified angle with the walls of the field.

It is not as robust or as general as it might be. There is no test to ensure that the point provided itself lies within the field, and it only deals with fields extending from X = 0 to XLIMIT and Y = 0 to YLIMIT.

PHASE III

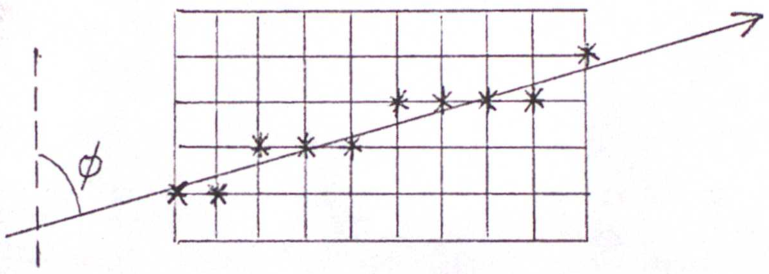
The processes described so far will convert the set of line segments produced into a network of chains, each end of each of which terminates at a domain boundary or on one of the other chains, but there is no guarantee that the chains do not cross each other, or that they do not enclose extremely small areas.

In order to remove such topological anomalies, we first convert to a grid point notation. This provides a straight-forward way of determining which side of a line a given point lies. No such simple process is readily apparent for lines stored as sets of points in a continuous field.

When working in grid point notation we use different axes, now the origin is in the top left corner and the +y direction is south. This conversion is effected before the topological rationalisation subroutine (TOPRAT) is called.

The conversion to grid point representation proceeds as follows:

- All the lines are labelled; for each segment of each chain we ascertain the direction in which the line is pointing, and then flag a sufficient density of grid points that unflagged points on either side of the line can never be orthogonally adjacent.



In this case, for example, the line is projecting in the +I direction, so for each value of I to be crossed, we calculate a real value for J, add 0.5 and then take the integer part (had the angle ϕ been less than $\pi/4$ we would take one point at each J value).

Chains of points are all labelled with even numbers, the magnitude of which gives the latitudiual tolerance of the line, on a scale such that 100 represents an expected error of one grid space.

- The patches between the chains are then filled in, using a different odd number for each patch.
- Tests are then performed to check for any topological anomalies and remove them
- The perimeter of each area is measured, and the value of $(\text{perimeter})^2/\text{area}$ calculated. For a circular region this would have the minimal value of 2π. If this parameter exceeds the empirically derived value of 35, the patch is considered unsatisfactory. Patches identified as being unshapely in this way are cut into two by a subroutine

called 'CLEAVE'. The cutting line is labelled with index 2 (4 is the smallest number allowed for points on fronts) and one of the two halves is relabelled with a new negative number.

- Very small areas, or small protruberances from large areas, are then identified with the aid of a subroutine called 'BUBBLE', which measures, for each point in the area, the largest circular space centred on that point which is entirely within the area (actually for ease of computing we use diamond-shaped spaces and not circles). Backwaters and small areas are then identified by their lack of proximity to any large 'bubble', and labelled by changing the sign of their indices.

Patches labelled with negative indices are removed by a subroutine called 'SCALPL', which also removes all the even numbered points (the ones on the fronts), replacing them with the largest adjacent odd number, leaving us with the field divided up into labelled patches, each labelled with a different odd number.

In the present version, no use is made during the last mentioned process of the information on expected accuracy of the fronts contained in their labels, this would, however be the appropriate point at which to do so.

We now describe in more detail the subroutines used during the topological rationalisation process.

<u>PERIM</u>

This is a subroutine to step round the perimeter of a patch and measure its length.

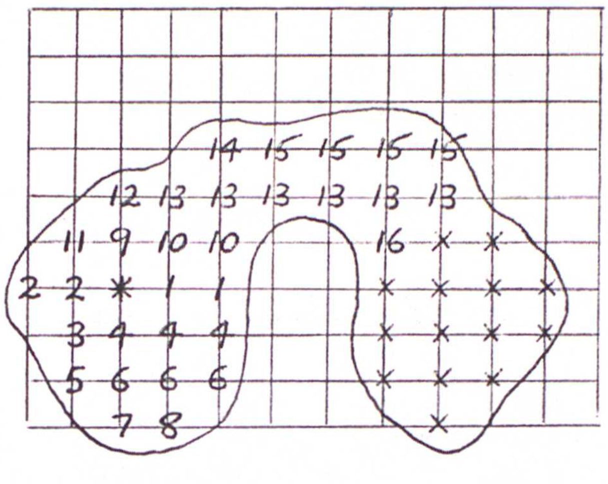
The input is a logical array, and it is the perimeter of the 'true' patch nearest to y = 0 which is measured.

The variable KAIM which takes a value 1, 2, 3 or 4 tells which direction we are heading in, the algorithm is simply to find a point on the boundary, flag it, then step round the boundary; at each step we attempt to turn right, if that puts us out of bounds then try straight ahead etc. When we return to the flag, the perimeter has been described.

<u>MOWER</u>

This subroutine is used to fill in the patches between fronts after the fronts have been labelled. It is provided with an index defining unlabelled points (which will usually be zero), and an index with which to label them, also with the coordinates of a point within the area.

The procedure works as follows (see diagram):



Starting at * we move to the right hand boundary (1), then to the left (2), labelling points as we go. Next, each of the points in the line below that just labelled is checked. This is done by adding the variable NORTH to the y coordinate, NORTH can be either +1 or -1.

If a point is found, as at (3), we proceed to complete that east-west line (4) and continue in like manner (5-8).

No more points can be found below (8) so we return to the start, alter the sign of NORTH, and move up the picture (9-15).

Often, as in this case, the area will not yet have been completely covered. Further unlabelled points are sought by stepping around the perimeter, exactly as in subroutine PERIM, seeking forgotten corners.

In this case, the first such point to be found will be (16), and heaving found it we re-initialize the process, and fill in all the points (X), after which the tour of the perimeter can be completed without further incident.

CLEAVE

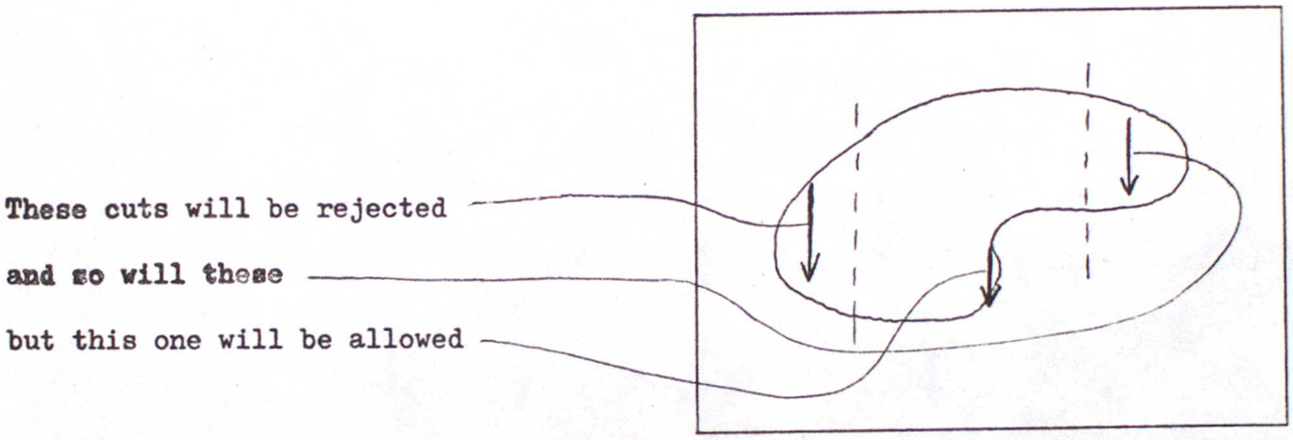
This subroutine again works on a logical array in which the area to be treated has been labelled as 'true'.

The object is to find a line along which the area can be cut in half, such that a significant portion of the area lies on each side of the lines but the length of the line is as short as possible, and no large open spaces within the area are cut.

In the current version the only cuts tested are those parallel with the X axis and with the Y axis. These are handled by separate sections of code, and it would be easy to add further sections testing cuts parallel with X = Y and X = -Y, though these do not appear to be necessary.

The process is to scan through the area, testing every possible cut, and using the following three constraints:

i. At least $^1/5$ and not more than $^4/5$ of the points must already have been visited. This is not quite precise, since in the case shown here:



In practice this does not matter much, since the small areas cut off will be put back again by subroutine SCALPL.

ii. The length of the cut is minimised.

iii. In cases where two cuts of equal length are acceptable, we run the 'BUBBLE' subroutine (actually this is done beforehand), add up the bubble indices of the points on the cut, and minimise that value.

The subroutine returns a code number indicating in which direction the cut runs, coordinates for the start, and the length of the cut. The actual cutting is done in the calling programme.

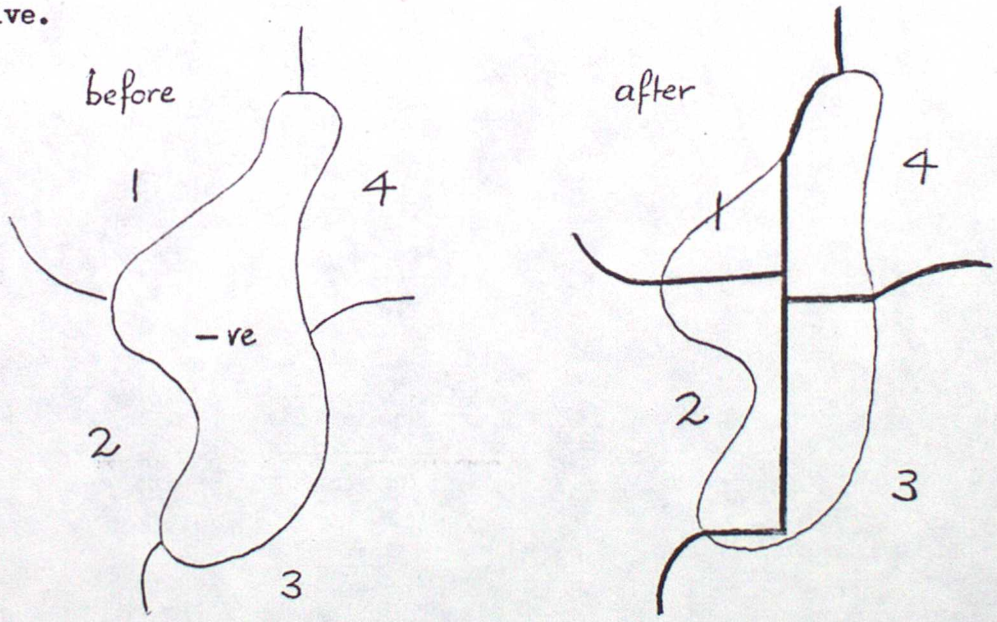
<u>SCALPL</u>

This subroutine replaces all areas not labelled with negative even numbers by some adjacent odd number, in such a way as to avoid generating small protruberances of one area into another.

Firstly, all the even numbers are removed (these are the dividing lines which had been necessary to keep the patches separate). This is done by replacing each even number by the largest adjacent odd number until no even numbers remain.

Next, patches of negative numbers are removed (these are the points which have already been flagged as unacceptable). When a patch of negative numbers is found, various parameters are measured, such as the number of different areas on which it impinges, whether it touches any walls, and its extreme values in X and Y directions. This classification is unnecessarily complex for the uses to which it is currently put, but has been retained so that a wider range of processes can easily be instituted to deal with different shaped negative areas should that prove useful.

The process used at present is, first to decide whether the negative area is longer in the X or Y direction, then to take a line just outside the patch on each side and move it inwards, replacing values on the line by those found whenever they are positive, and replacing the number found by the one on the line when it is negative.



In cases where the negative area touches a wall, the line is only moved in from the other side.

Several aspects of this subroutine could be refined. Most notably, the information on tolerance of the positions of the fronts could be used. Those sections of front which have been extrapolated to the boundaries in order to make them separate off the areas on either side of them tend to have extremely low tolerances, and should really be the first to be adjusted in removing topological anomalies.

POTENTIAL IMPROVEMENTS

The following brief comments are intended to describe the outstanding weaknesses and problems of the programmes described above, with a few remarks as to how they might be remedied.

(a) The weakest section is the generation of segments of front from meteorological data. The output of the routines currently used does not accord well with intuitive ideas of what they should produce, and are not able to produce anything which the later stages of the programme can digest. There are probably bugs in these subroutines, but in any case they may need augmenting with information from θ_w discontinuities in radiosonde data or dewpoint changes at the surface. Further testing and refinement of these routines is the most urgent priority.

(b) Having generated some sensible frontal segments, the various timing parameters in subroutine TESS should be varied systematically in order to understand their effect. For this purpose it would probably be useful to have a CALCOMP routine drawing the fronts at intermediate stages.

(c) The coordinates, length units, axes and boundaries are all a bit arbitrary. This probably does not matter much for testing and development, but would need redoing before the programme could be used.

(d) Subroutine SCALPL could usefully be extended in two ways:

i. To incorporate the information on tolerance on position of fronts in correcting their position to remove 'warts'.

ii. To allow points on fronts to remain in the final field where the confidence is high, in order to be able to introduce a special deep convection scheme on active fronts.

APPENDIX

The following Fortran programmes are, at the time of writing, stored as separate members on M11.ZSRCELIB. All those not beginning with the letter E are stored with an E preceding their names.

EXUME	extracts meteorological data (only for last 48 hours)
EWER	prints out those data sets
ECHO	cleans and copies them
EPIC	runs everything else, calling WVEER, BARKIK, TROUGH and TESS
WVEER	finds stations where the wind has veered
BARKIK	finds stations where the bar has kicked
TROUGH	finds troughs between pairs of stations
TESS	Tesselates the field, calling SEGFIT, COWCAT NODES, WALLPT and TOPRAT
SEGFIT	calculates the fit between two segments
CONCAT	concatenates many segments into one
NODES	finds where a line crosses a front
WALLPT	finds where a line hits the wall
TOPRAT	Rationalises the topology, calling KPRINT, MOWER, POINTS, PERIM, BUBBLE, CLEAVE and SCALPL.
KPRINT	prints the grid at some intermediate stage
MOWER	visits and flags all points in a connected patch
POINTS	turns a vector into a compass bearing
PERIM	measures the perimeter of a patch
BUBBLE	determines how big a bubble you can blow
CLEAVE	cuts a patch in half
SCALPL	cuts off warts
EXAM	is a separate main programme to run TESS on artificial data
ERAT	is a separate main programme to run TOPRAT on artificial data.

The following JCL data sets reside on T11YY.JCL.CNTL:

ECHO	runs Fortran programme ECHO
EXUME	" " " EXUME

EWER	"	"	"	EWER
EPIC	"	"	"	EPIC using datasets created by EXUME
ETJCL	runs Fortran programme			EXAM using data set on T11YY.DATA.DATA (ETDATA)
ERAT	runs Fortran programme			ERAT using data set on T11YY.DATA.DATA (ERAT).

The following data sets on SYS005 were created by EXUME on 6 March 1981:

M11.E MARØ6Ø5 ; M11.EMARØ6Ø6