Matrix of derivatives d eqn(i)/d var(j)

eqn\var	f	T	m	X	r	L	f	T	m	X	r	L	f	T	m	X	r	L	C		
P(f)=g(r)	*		.		+				.						.						2
L=r^2T^4		*	.		+	+			.						.					ZSBC	3
mdot=f(T)		+	*	.					.						.						1
h(f)X'=0	+			.	*		+			.	*				.					FSBC	6
r'=C/d(f)	+			.	*		+			.	*				.				+		2
L'=Ce(T)		+		.		*		+		.		*			.				+		4
P'=Cg(r)	*			.	+		*			.	+				.				+	FDE	1
T'=CL		*		.		+		*		.		+			.				+		3
m'=C			*	.					*	.					.				*		5
X''=R(T)		+		.	*			+		.	*			+		.	*			SDE	6
r'=C/d(f)				.			+			.	*		+			.	*		+		2
L'=Ce(T)				.				+		.		*		+		.		*	+		4
P'=Cg(r)				.			*			.	+		*			.	+		+	FDE	1
T'=CL				.				*		.		+		*		.		+	+		3
m'=C				.					*	.					*	.			*		5
h(f)X'=0				.			+			.	*		+			.	*			FCBC	4
r=0				.						.						.	*				3
L=0				.						.						.		*		ZCBC	2
m=0				.						.					*	.					1
	f	T	m	X	r	L	f	T	m	X	r	L	f	T	m	X	r	L	C		
	1	2	4	5	7	8	1	2	4	5	7	8	1	2	4	5	7	8	6		

For illustrative purposes, the system of equations has been trivialised down to (a) 6 Variables, f, ..., L (b) 3 meshpoints (surface at left, centre towards right), (c) 1 Evalue, C, at far right (d) 3 Zero-order Surface BCs, first 3 rows, (e) 1 First-order SBC, 4th row (f) 5 First-order DEs (g) 1 Second- order DE (h) 5 FDEs again (i) 1 First-order Central BC (j) 4 Zero-order Central BCs. This gives 19 equations in 19 variables.

The matrix has a block tri-diagonal structure. The variables and the equations have been ordered (ie permuted relative to their numbering in the code, which is indicated along the bottom, and down the RH side) so that very significant terms are on the leading diagonal -- except for the bottom RH element. However this element gets a significant value in the course of the elimination of the material below the leading diagonal (see later).

```
   V   .   .   .   .   V   E   (actual)
   1   2   4   5   7   8   6   (1245876)

 FSBC
  SDE  FDE  .   .   .  FDE
   6   2   4   1   3   5       (642135)

 FCBC  ZCBC . ZCBC  ZSBC . ZSBC
   4   3   2   1   2   3   1   (4231231)
```

(continued on next page)

(continued from previous page)

The above rows give the values for the array IE on p10 of the writeup, except that
(a) I 'misremembered' the ordering of L, r and consequential on that the ordering of the L', r' FDEs, and the L, r ZCBCs. This permutation doesn't matter at all.
(b) The real code has 5 composition Variables instead of 1 (X). The 4 extra composition Variables are numbered 3, 9, 10, 11. The 4 extra SDEs are numbered 7, 8, 9, 10, as are the 4 extra FSBCs. The 4 extra FCBCs are numbered 5, 6, 7, 8.

The 4 below m means that the m-Variable is stored as H(4,K) at meshpoint K. The 3 beside ZSBC means that in the ZSBC portion of subroutine EQUNS,

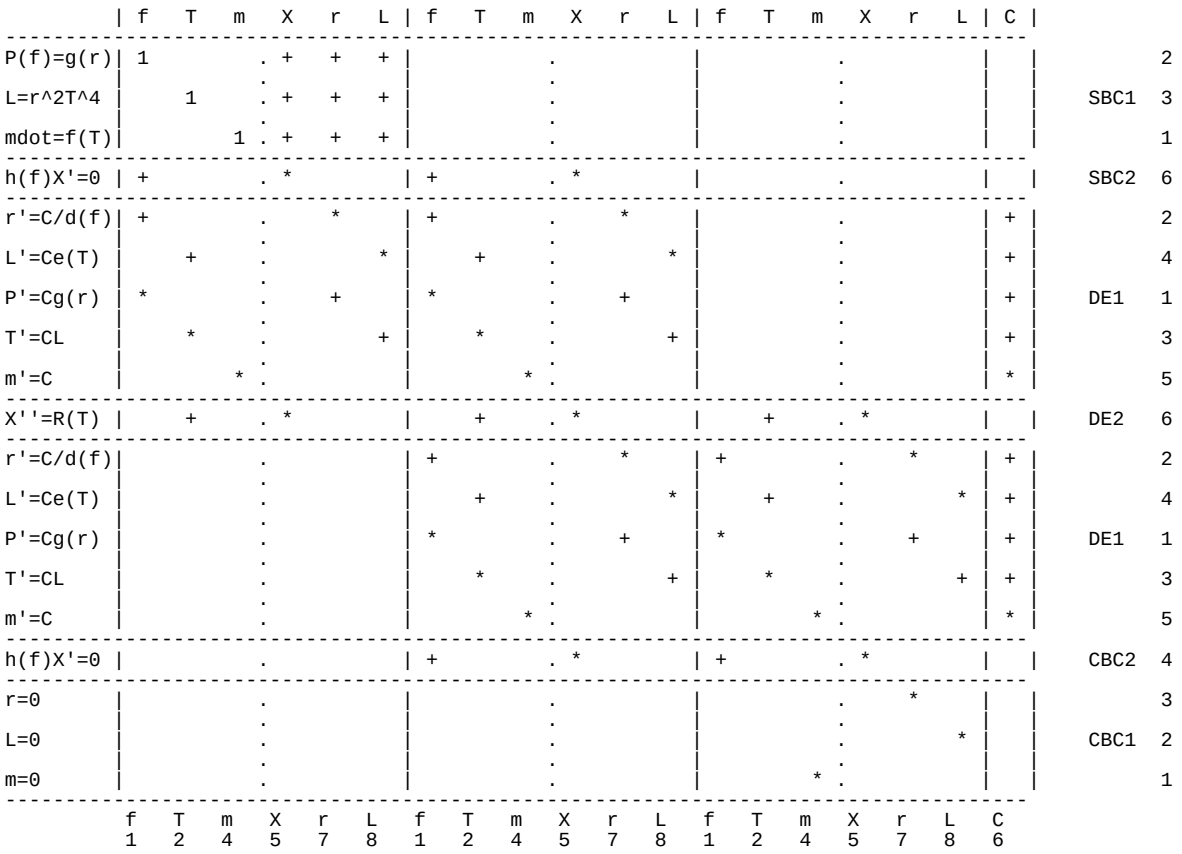
EQU(3)=L-r^2T^4.

The 1 beside FDE means that in the FDE portion of EQUNS,

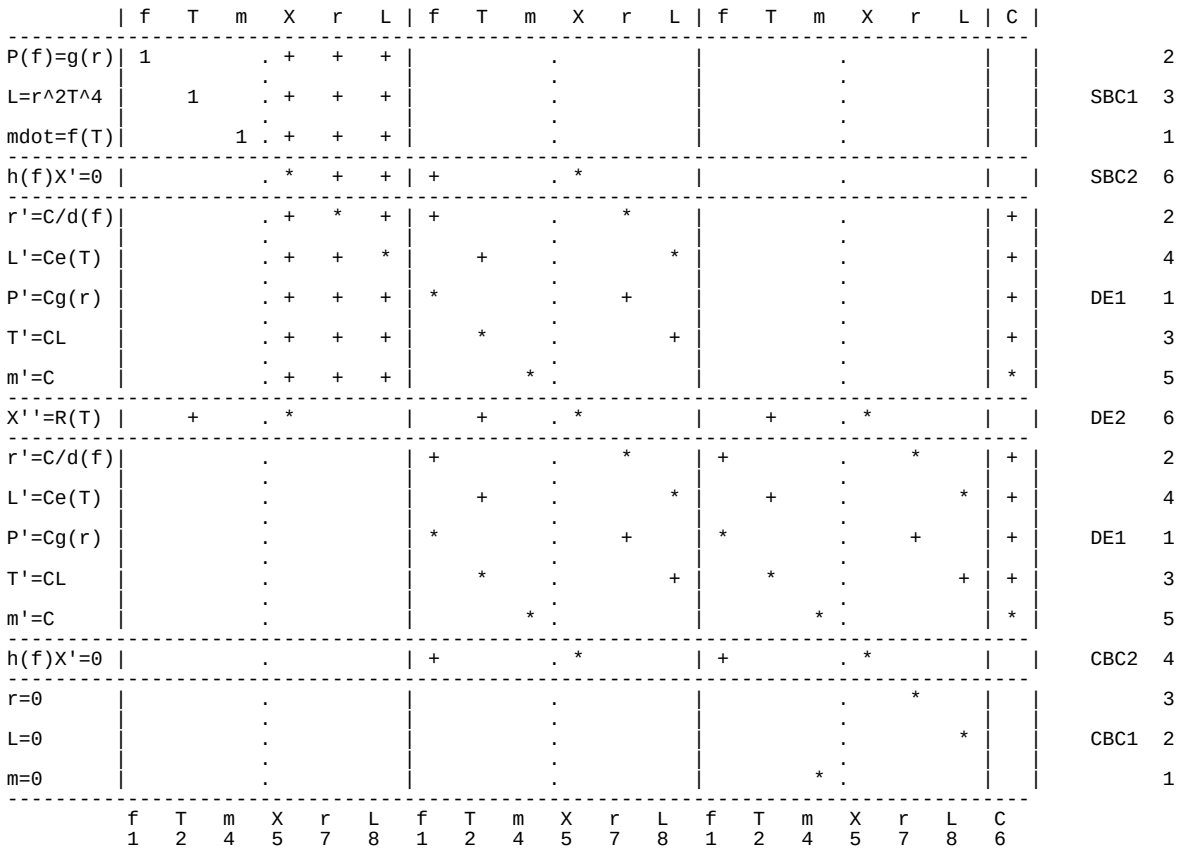
EQU(1)=P(K)-P(K-1)-0.5C(g(K)+g(K-1))

and so on. In the illustration, P is assumed to be a function of f only, rather than of f, T, and its gradient P' to be C times a function of r only, etc.

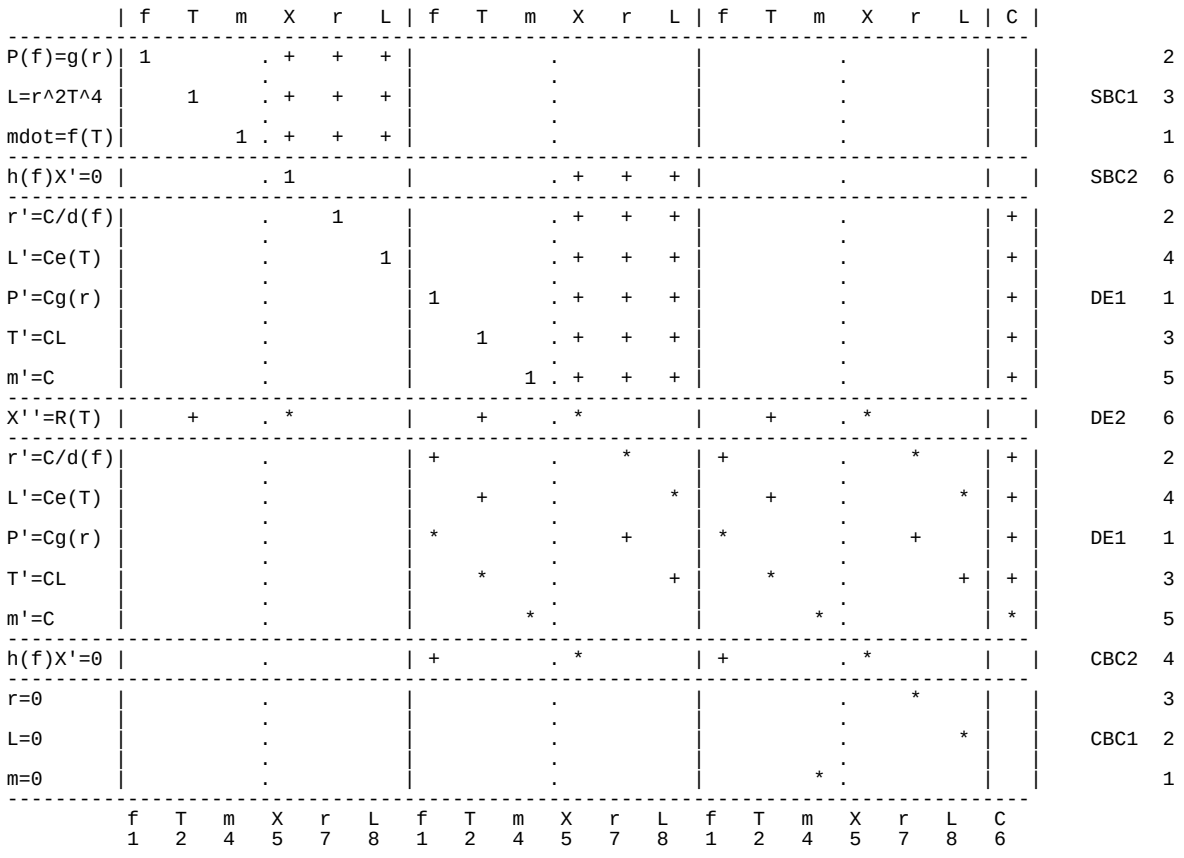
* means a significant term; + is a minor term, might be comparable but need not be. Every block that contains either *s or +s also in practice contains more non-zero but unimportant terms; all blocks that have neither are completely empty, except for the 3rd and 4th block in the bottom row. Since the ‘central' meshpoint is actually half a meshpoint in from the centre (to avoid singularities like m/r^2), the real ZCBCs involve *both* innermost meshpoints.



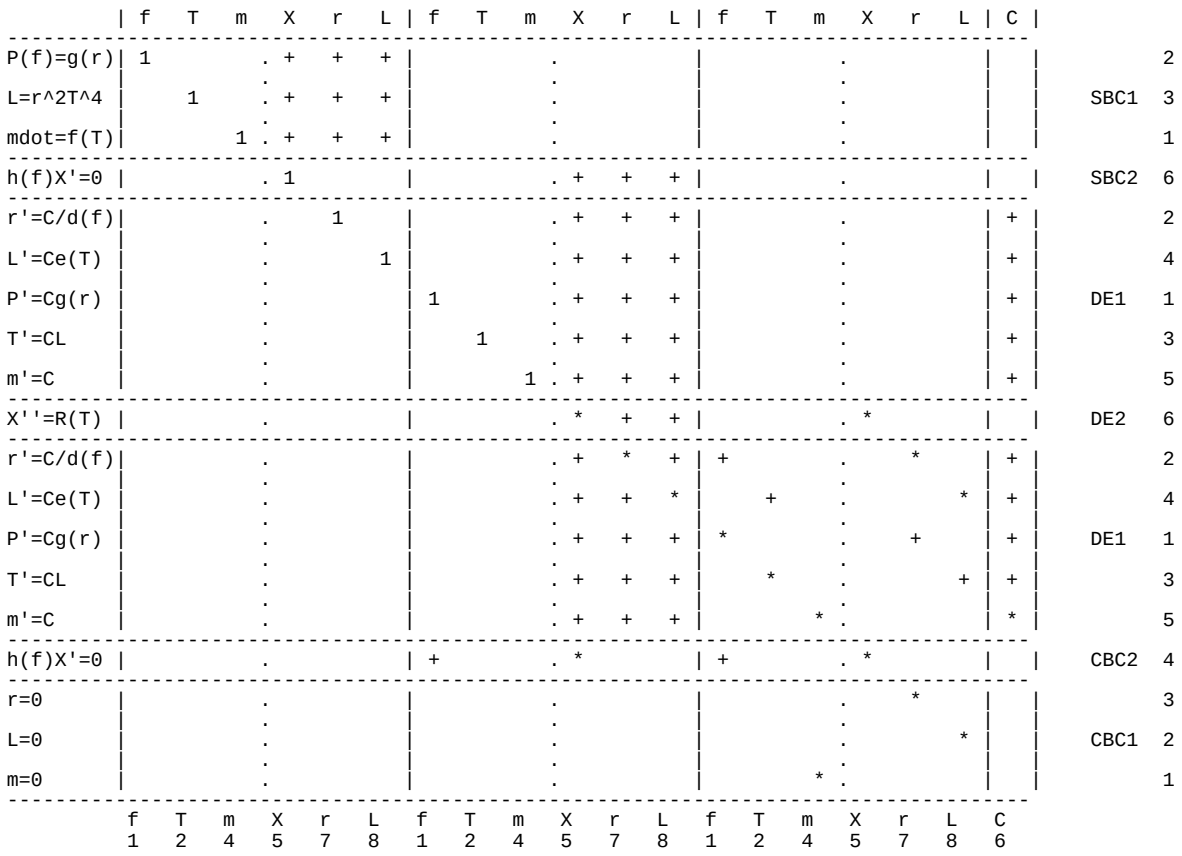
Successive calls to DIFRNS in SOLVER set up successive rows of blocks in the matrix. In between these calls, Gaussian elimination proceeds via calls to ELIMN8 and DIVIDE.
The first call to DIVIDE mutiplies the top 3 rows by the inverse of the leading 3 X 3 block, reducing this block to unity. The block to its immediate left is stored (and also the columnn on the far right, which is not illustrated: when operating on the array A in Ax = y, the same operatioins are done on y as on A, but we only illustrate A).
After the first call to DIVIDE, we call DIFRNS again to fill the next 6 rows (SBC2 and DE1). Then a call to ELIMN8 eliminates the blocks below the leading unit matrix: see next Fig.



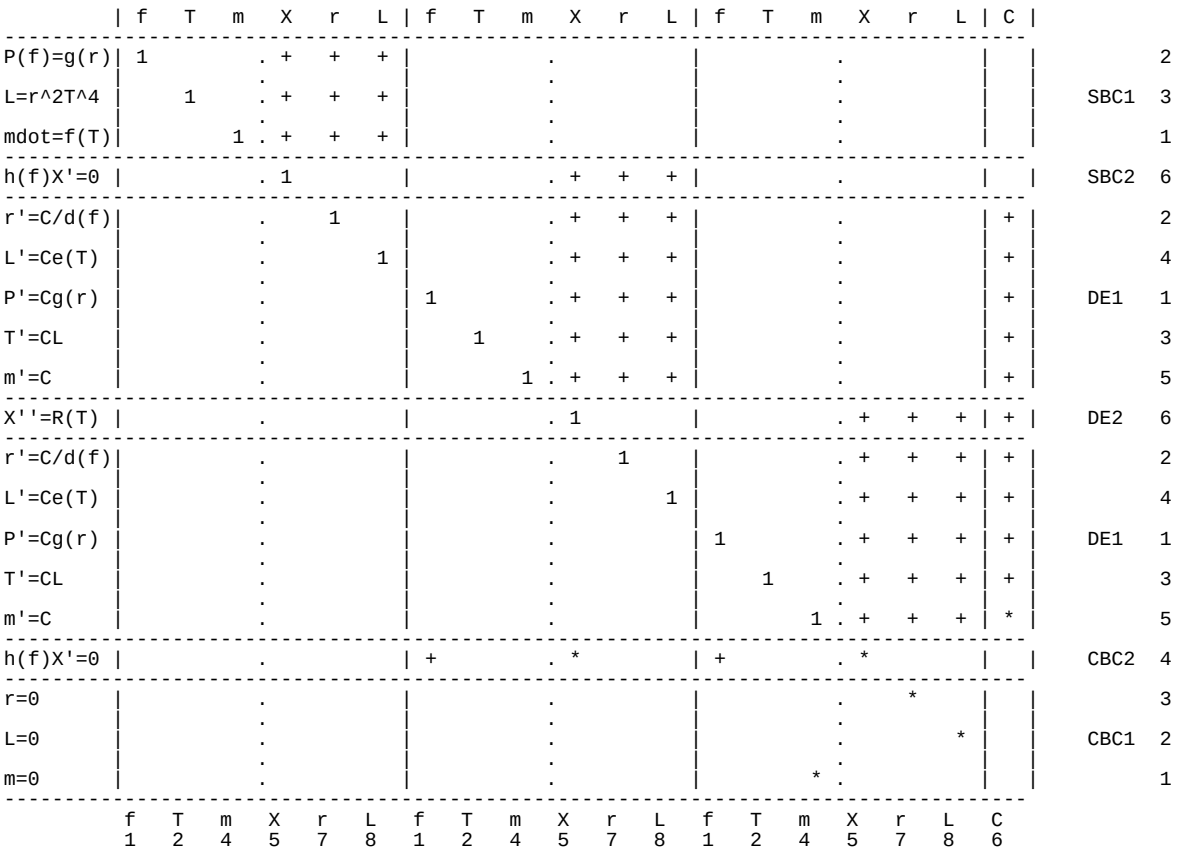
Then in DIVIDE we divide the current 6 rows (SBC2 and DE1) by the leading 6 X 6 matrix: next page.



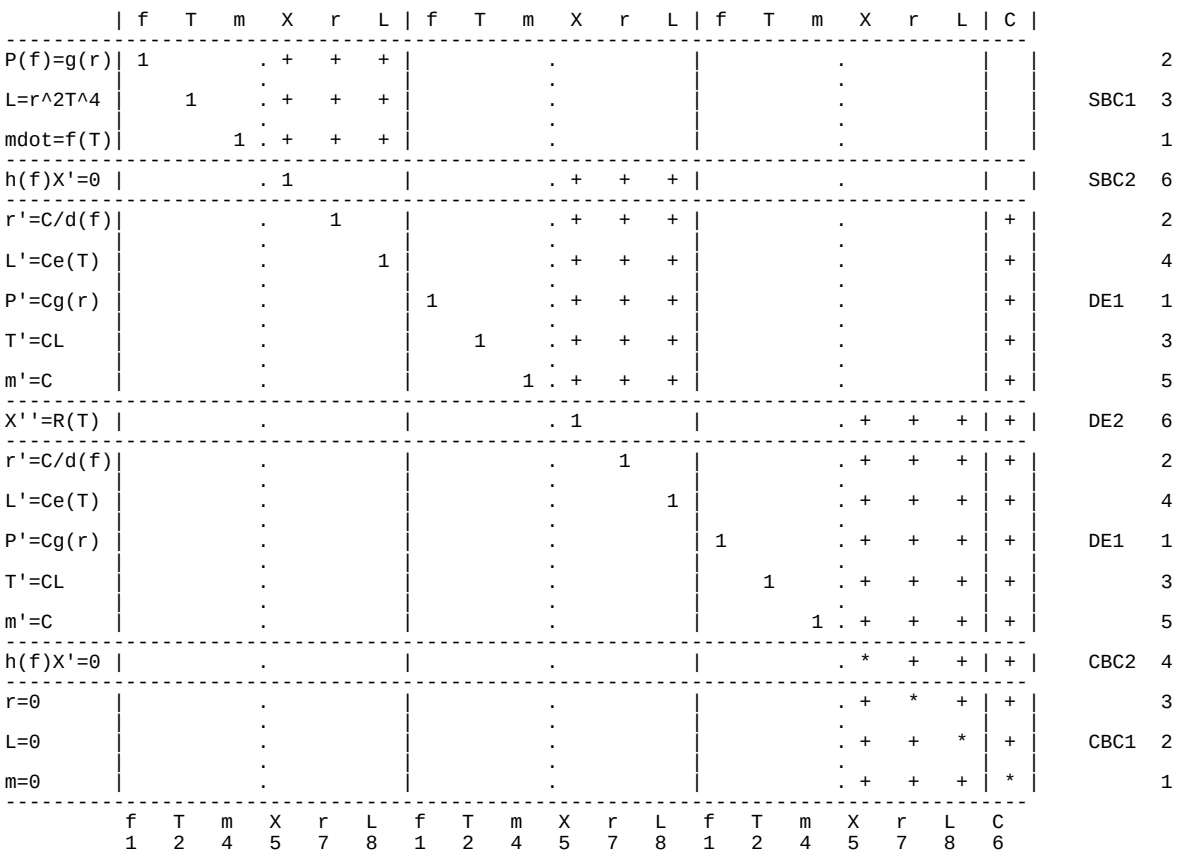
Next, we fill up the next 6 rows (DE2, DE1) in DIFRNS. Then we use ELIMN8 to eliminate everything below that part of the leading diagonal which is already reduced to unit matrices. See next fig.



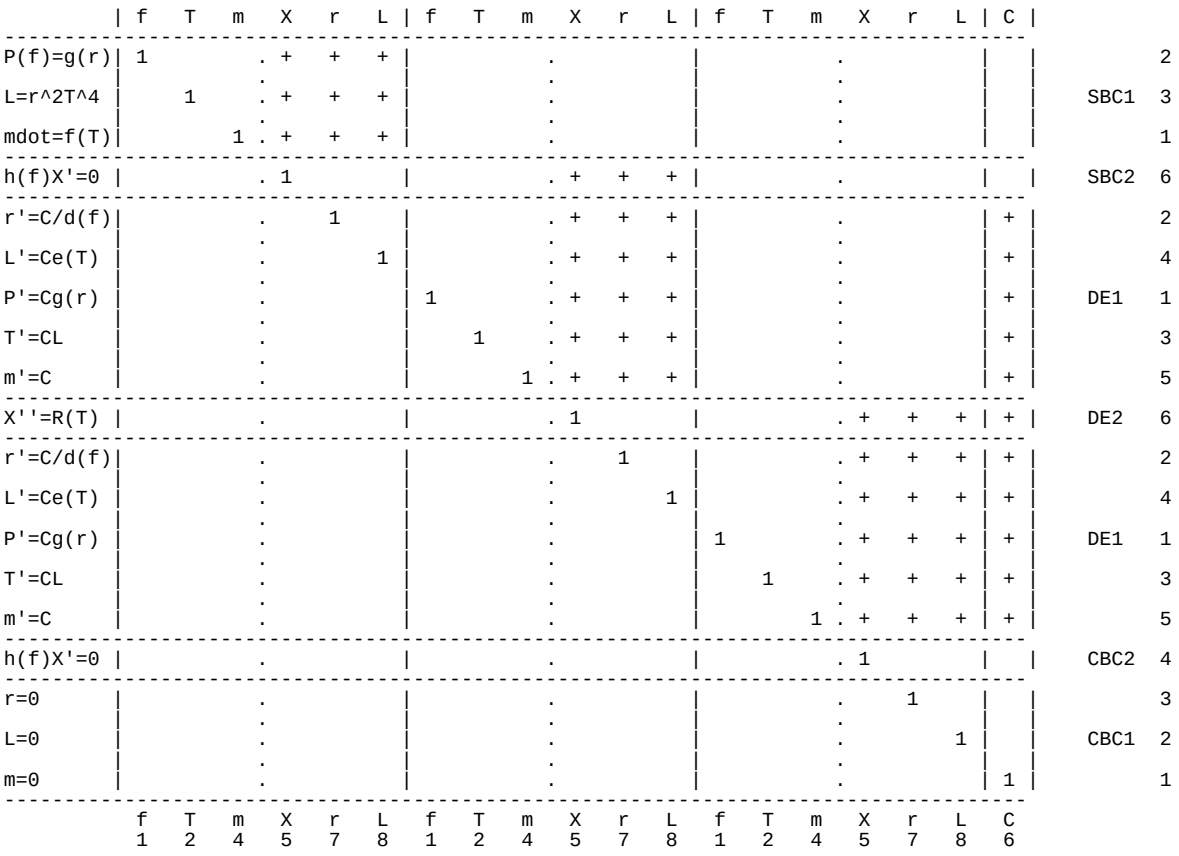
Now use DIVIDE to convert the leading 6 X 6 array of rows DE2, DE1 to unity: next fig.



Now fill up the last 4 rows (CBC2, CBC1) in DIFRNS, and ELIMN8 what is below the unit part of the leading diagonal: next fig.



This must have the effect of putting some non-trivial value in the bottom RH corner, which was previously empty. In fact, the previous ELIMN8 step will put there the product of two starred values -- one at the far right of the last m'=C row, and one in the last (m=0) row -- in the previous fig. This should leave a reasonably invertible 4 X 4 block in the bottom right: next fig.



We can now proceed by back-substitution to obtain the required dC, {dL, dr, ... df at central meshpoint}, {dL, dr, ... df at intermediate meshpoint} and finally {dL, dr, ... df at surface meshpoint}.

It should be clear that while some permutations of variables and/or equations have a trivial effect -- eg interchanging r and L, or f and T -- others permutations can have a completely disruptive effect -- eg interchanging m and X. Determining a viable permutation is as much an art as a science, in my experience.

Suppose we wish to add a few extra equations, as I did recently: the moment of inertia I such that

$$I' = 0.667\ mr^2C$$

(remembering that C = m' = dm/dk), and Prot, Horb and e: the last 2 are the orbital angular momentum and the eccentricity The extra BCs are I=0 at the centre, and at the surface

d(Horb+2.pi.I/Prot) = ..., dHorb/dt = ..., de/dt=... .

(continued on next page)

(continued from previous page)

The unspecified derivatives involve tidal friction, etc. We have apparently 1 new Variable (I) and 3 new Evalues (Prot, Horb, e). There would be KM+3 new equations (KM = no. of meshpoints, = 3 in the illustration): KM-1 DE1s (for I) and 4 BCs, ie 1 CBC1 and 3 SBC1s. This looks OK, but apparently it isn't -- the 3 new Evalues would contribute elements at the *RH* end of the top row of blocks, but the the 3 new SBC1s would require some non-trivial elements at the *LH* end, in order to give an invertible 6 X 6 block at the top left.

I have found it necessary to treat only Prot as an Evalue, and to treat the other 2 quantities (Horb, e) as spurious new Variables, satisfying

Horb' = 0, e' = 0.

One can squeeze in 3 new Variables (I, Horb, e) along the top, between m and X at each meshpoint, the new Evalue at the end, after C, the 3 new SBC1s after mdot, the 3 new DE1s after m' = C, and the 1 new CBC1 after m = 0. This allows the leading diagonal matrices to be invertible.

The moral seems to be that it would have been better to write the code so that the centre of the star is at the top left and the surface at the bottom right. Apparently one should start at the end which has the *fewest* BCs; this is exactly the opposite of what I supposed in 1969 when I wrote the code. It should be possible to rewrite the solution package to work from centre to surface, but my heart sinks at the thought of it. Besides, one might wonder whether the saving of cpu time is worth it. The current version requires inverting 9 X 9 matrices, apart from the first and last; revised as I suggest this would be reduced to 7 X 7. But actually the figures are 13 X 13 and 11 X 11, since I have ignored 4 of the 5 composition equations. The cpu time goes as the cube of the size of the matrix.