COMPUTATION OF AGE-DEPENDENT RELAXATION SPECTRA

Zdeněk P. Bažant and Ali Asghari
Department of Civil Engineering
Northwestern University
Evanston, Illinois 60201, U.S.A.

(Communicated by F. H. Wittmann)
(Received March 15, 1974; in final form April 1, 1974)

ABSTRACT

A computer algorithm for the computation of discrete age-dependent relaxation spectra of concrete from creep or relaxation test data is presented and a full FORTRAN IV listing is given. The algorithm is based on a previously outlined method, consisting in the expansion of relaxation curves in series of real exponentials on the basis of a least square criterion. This method is refined herein by imposing suitable smoothing conditions upon the spectra in order to reduce spurious sensitivity of results to small changes in given creep data. Two variants are given; in one the spectrum is approximated as a cubic polynomial in the logarithm of age and the logarithm of the relaxation time, while in the other one a cubic polynomial in the logarithm of age alone is used. Numerical examples show that given smooth creep data can be recovered from the spectra with a negligible error, which demonstrates that the spectra fully characterize creep properties. Apart from being a fundamental characteristic of creep, the relaxation spectra convert a hereditary creep law into a history independent rate-type form, which is requisite for creep analysis of large structural systems.

Un algorithme pour le calcul à l'ordinateur des spectres de relaxation du béton à des âges différents est présenté, avec un programme en FORTRAN IV. Les courbes de relaxations sont exprimées en série d'exponentielles, utilisant la méthode des plus petits carrés et imposant une condition de courbure minimum des spectres, ce qui réduit la sensibilité aux petites variations des données sur le fluage. Les modules de relaxation sont exprimés en forme de polynômes cubiques de l'âge. Les exemples numériques montrent que les courbes de fluage données peuvent être obtenues à partir des spectres. Les spectres transforment un loi héréditaire linéaire de fluage en équations différentielles, ce qui est utile pour le calcul des constructions à grand nombre d'inconnues.
Introduction

In the theory of time-decaying phenomena, the relaxation spectra play an equally fundamental role as the frequency response spectra do for the periodic phenomena. This has been recognized long ago in the field of viscoelasticity, and the relaxation spectra of linearly viscoelastic materials have been extensively studied (1-3). However, all of the classical works have been confined to time-independent viscoelasticity, which is applicable to polymers but not to concrete, except when the hydration process ceases as in the case of low temperatures, completely dried samples, or completely hydrated samples. Relaxation spectra of materials with age-dependent properties have been recently formulated (4,5) and a method of their determination has been outlined.

The aim of this paper is to present a detailed FORTRAN IV listing of the computational algorithm, in order to facilitate the practical use. In addition, the method from (5) will be refined by introducing more appropriate smoothing conditions which reduce the spurious sensitivity of the computed spectrum to small changes in given creep data.

Review of the Basic Formulation

With the well known limitations and certain simplifying assumptions (4,5) the creep law of aging concrete may be assumed to be linear, satisfying the principle of superposition. Then it can be described with any desired accuracy by a sufficiently large system of first-order linear differential equations whose structure can be conveniently visualized by the Maxwell chain model (Fig. 1). Since generalization to multiaxial stress is straightforward (6), attention will be restricted to uniaxial stress. The stress-strain law based on Maxwell chain is then expressed in the form (4,5)

\[ \sigma = \sum_{\mu=1}^{n} \sigma_{\mu}, \quad \dot{\varepsilon} = (\dot{\sigma}_{\mu} + \sigma_{\mu}/\tau_{\mu})/E_{\mu}(t) \quad (\mu = 1, \ldots, n) \]

in which \( \sigma = \) stress, \( \varepsilon = \) strain, \( \sigma_{\mu} = \) stress in \( \mu \)-th spring in Fig. 1, called
hidden stress (or internal variable), \( \tau_\mu \) = relaxation time associated with the \( \mu \)-th spring, \( E_\mu \) = elastic modulus of the \( \mu \)-th spring, depending on time \( t \) which represents the age of concrete. Superimposed dots represent time derivatives, e.g. \( \dot{\varepsilon} = d\varepsilon/dt \). Shrinkage strain is not included.

Eq. (1) belongs to the class of the so-called rate-type creep laws, which have recently appeared to be preferable to integral-type creep laws involving hereditary integrals, for several reasons. First, thermodynamics of time-dependent phenomena is much simpler in terms of current values of state variables, including the hidden variables \( (\sigma_\mu) \), than it is in terms of the time-histories of stress and strain. Second, in numerical analysis of large structural systems, the storage space in computer memory which is required for the rate-type creep laws is only a fraction of that required for integral-type laws. Third, a rate-type creep law allows establishing a correlation with the physical theory of the processes in the microstructure, which can provide valuable additional information (e.g., on the form in which temperature and water content should appear in the creep law (7,8)).

The plot of \( E_\mu \) versus \( \log \tau_\mu \), which is discrete and is called relaxation spectrum, depends on age \( t \). The actual spectrum of the material is, of course, essentially continuous, and the \( E_\mu \)-values are its approximate discrete representation. Taking this point of view, it becomes clear that the continuous spectrum can be approximated equally well for any set of \( \tau_\mu \)-values which are sufficiently densely distributed in log-time over the whole range of interest (e.g., the spectrum given by dashed lines in Fig. 1 is equivalent). Therefore, one should not try to determine \( \tau_\mu \) from the creep data (and indeed, if attempted, a problem with a non-unique solution would result (11)). Rather, the \( \tau_\mu \)-values must be chosen. In particular they may be chosen time-constant and a suitable choice is (5)

\[
\tau_\mu = \tau_1 10^{\mu-1} \quad (\mu = 1, \ldots, n-1), \quad \tau_n = \infty \quad (\mu = n)
\]

(2)

where the last, infinite relaxation time is a convenient form of expressing the fact that the last spring in the chain has no dashpot attached to it.

Eqs. (1) may be easily integrated when \( \varepsilon(t) \) is a step function, i.e., \( \varepsilon = 1 \) for \( t \geq t' \), \( \varepsilon = 0 \) for \( t < t' \). The corresponding \( \sigma \) represents the relaxation function \( E_R(t,t') \) and has the form

\[
E_R(t,t') = \sum_{\mu=1}^{n} E_\mu(t') e^{-(t-t')/\tau_\mu}
\]

(3)
This is a series of real exponentials, called Dirichlet series. It is a counterpart of the series of imaginary exponentials, or Fourier series, arising in the study of periodic phenomena.

The time-dependent properties of concrete are usually characterized in terms of the creep function, \( J(t,t') \), which represents strain \( \varepsilon \) at time \( t \) caused by a constant unit stress acting since time \( t' \). The relaxation function, \( E_R(t,t') \), may be computed from given \( J(t,t') \) by an algorithm which was presented in (9). Later a more efficient version of this algorithm was proposed in (10). A FORTRAN IV subroutine RELAX, based on (10), is presented in the Appendix, because a conversion of \( J(t,t') \) into \( E_R(t,t') \) is a necessary step toward determining the relaxation spectra, unless sufficient data from stress relaxation tests are available.

Function \( E_R(t,t') \) which describes given test data is numerically characterized by discrete values \( E_{R_{\alpha}} = E_R(t'_\alpha + t_{r\alpha} + t') \), where \( t'_\alpha \) = chosen discrete ages at the start of relaxation, \( t_{r\alpha} \) = chosen discrete times elapsed from the start of relaxation. These discrete values must be sufficiently densely distributed in both log \( t'_\alpha \)- and log \( t_{r\alpha} \)-scales. A uniform distribution with three \( t_{r\alpha} \)-values per log 10 and two \( t'_\alpha \)-values per log 10 is normally sufficient. Both \( t'_\alpha \) and \( t_{r\alpha} \) must completely cover the whole time range of interest.

Using the method of least squares, the values \( E_{u_{\alpha}} = E_{\mu}(t'_{\alpha}) \) (for \( u = 1,\ldots,n; \alpha = 1,2,\ldots,n_{\alpha} \)) may be found by minimizing the expression

\[
\phi = \sum_{\alpha} \sum_{r} \left( \sum_{\mu=1}^{n} E_{\mu} e^{-t_{r\alpha}/t_{\mu} - E_{R_{r\alpha}}} \right)^2 = \text{Min.} \quad (4)
\]

However, fitting of given data by Dirichlet series appears to be much more difficult than it is in the case of Fourier series (11). The difficulties are due to the fact that an ill-conditioned system of linear equations arises and that very different \( E_{\mu} \)-values can represent the same data nearly equally well, i.e., coefficients \( E_{\mu} \) are unstable functions of the data.

Identification of Relaxation Moduli from Given Relaxation Data

Method 1. Improved Version of the Method from Ref. (5)

The afore-mentioned numerical difficulties have been circumvented in (5) by imposing the condition that the \( E_{\mu} \)-values as functions of \( \mu \) must be smooth, which seems to be a physically plausible requirement. Using the least square method, this is mathematically expressed by adding to \( \phi \) in Eq. (4) a sum of squares of the first and second differences of \( E_{\mu} \) as functions of \( \mu \); see
Eq. 8 from Ref. (5). In computing the fits of the extensive typical creep data in (5) it was tacitly assumed that $E_\mu$ had to be a smooth function of $\mu$ up to the last spring $\mu = n$ which is attached to no dashpot ($\tau_n = \infty$). Although close fits of data were obtained in (5), it has later been found that this assumption was unjustified and was responsible for the fact that similar creep data yielded very different $E_\mu$-values (see Table 2 of (5)). Correctly, the last spring modulus, $E_\mu = E_n$, can have no relationship to the preceding spring moduli $E_{n-1}$, $E_{n-2}$, etc., for the following reason. First one should note that for the elapsed times $t - t'$ that are less than $\tau_/10$, the displacement that occurs in the $\mu$-th dashpot, as well as in the dashpots number $\mu + 1$, $\mu + 2$, etc., is insignificant and the dashpots are equivalent to rigid coupling. The actual continuous relaxation spectrum normally extends far beyond the range of interest (Fig. 1), but all the dashpots to the right of this range may be viewed as rigid couplings. Thus, the set of springs to the right of the range is equivalent to a single spring (Fig. 1), and the last spring modulus in the chain, $E_n$, should be regarded as a sum of the moduli of all springs to the right of the $(n - 1)$st spring in the actual model for an unlimited time range (Fig. 1). Hence, no smooth change from $E_{n-1}$ to $E_n$ may be enforced. (Note that on the left of Maxwell chain the situation is different; the stress in all dashpots and springs to the left of the range of interest is virtually zero and therefore all springs to the left of the range may be neglected.)

For other details the discussion in Ref. (5) is valid. A FORTRAN IV subroutine for this method, called MAXWLL, is provided in the Appendix, and ample comments within the program are inserted for clarity. After computing all $E_\mu$-values, the dependence of $E_\mu (t')$ upon log $t'$ is fitted by a cubic polynomial:

$$E_\mu (t') = C_1^\mu + C_2^\mu x + C_3^\mu x^2 + C_4^\mu x^3, \quad x = \log t'. \quad (5)$$

Method 2. Direct Identification of a Polynomial for Relaxation Moduli

The condition of smooth dependence of $E_\mu$ upon $\mu$ may be also enforced if $E_\mu$ is assumed in the form of a low-degree polynomial. A cubic polynomial appears to be suitable for concrete in the time range of $t - t'$ from 0.001 day to 10,000 days. A cubic polynomial is also suitable for the dependence on log $t'$, so that $E_\mu$ may be assumed as full two-dimensional cubic polynomial,
\[ E_{u}(t') = a_1 + a_2 \mu + a_3 \mu^2 + a_4 \mu^3 + a_5 \mu^4 + a_6 \mu^5 + a_7 \mu^6 + a_8 \mu^7 + a_9 \mu^8 + a_{10} \mu^9 \]

\[ = \sum_{j=1}^{10} a_j f_j(\mu, x), \quad x = \log t' \quad (\text{for } \mu < n). \]  

(6)

According to the preceding discussion, this polynomial may not be extended to the last spring of the chain, \( \mu = n \).

From experience it has been found that the first discrete time \( t_1 \) from loading should be about 0.2 \( \tau_1 \), and the last one, \( t_N \), about \( \tau_{n-1} \). Setting \( t_N = \tau_{n-1} \) exactly, Eq. (3) for \( t = t' + t_N \) yields

\[ E_{R}(t' + t_N, t') = E_{R_{N\alpha}} = E_{m\alpha} e^{-1} + E_{n}, \quad (m = n - 1) \]  

(7)

where all terms of the sum but the last two have been neglected since the exponentials have values less than \( e^{-10} = 0.000045 \). Eq. (7) yields

\[ E_{n}(t') = E_{R_{N\alpha}} - E_{m \alpha} e^{-1} \quad (m = n - 1). \]  

(8)

This provides a relation between \( E_{n} \) and \( E_{n-1} \) that should be satisfied a priori. Thus the end values \( E_{R_{N\alpha}} \) of stress relaxation are related to the last two spring moduli \( E_{n} \) and \( E_{n-1} \) and, in conformity with Eq. (6), \( E_{R_{N\alpha}} \) should be approximated as a cubic polynomial

\[ E_{R_{N\alpha}} = b_1 + b_2 x_{\alpha} + b_3 x_{\alpha}^2 + b_4 x_{\alpha}^3, \quad x_{\alpha} = \log t'. \]  

(9)

Now, expressing \( E_{n} \) by means of \( E_{m} \) and \( E_{R_{N\alpha}} \) from Eqs. (8) and (7), and substituting expression (6) into Eq. (1), one obtains

\[ E_{R\alpha} = \sum_{\mu=1}^{m} E_{\mu \alpha} \phi_{\mu \alpha}^{r} + \sum_{k=1}^{4} b_{k} x_{\alpha}^{k-1} \]  

(10)

where \( \phi_{\mu \alpha}^{r} = e^{-\tau_r/\tau_{\mu}} \) for \( \mu < m \), \( \phi_{m \alpha}^{r} = e^{-\tau_r/\tau_{m}} - e^{-1} \) for \( \mu = m \).

(11)

Insertion of (6) into (10) further provides

\[ E_{R\alpha} = \sum_{\mu=1}^{m} \sum_{j=1}^{10} a_{j} H_{\alpha j}^{r} + \sum_{k=1}^{4} b_{k} x_{\alpha}^{k-1} \]  

(12)

where \( H_{\alpha j}^{r} = \sum_{\mu=1}^{m} f_{j}(\mu, x_{\alpha}) \phi_{\mu \alpha}^{r} \).

(13)

Substituting Eq. (9) into the least-square expression (4) furnishes

\[ \phi = \sum_{\alpha r j=1}^{10} \left( \sum_{\mu=1}^{m} H_{\alpha j}^{r} a_{j} + E_{R_{N\alpha}} - \sum_{k=1}^{4} b_{k} x_{\alpha}^{k-1} \right)^2 + \sum_{j=1}^{10} w_{j} a_{j}^2 = \text{Min} \]  

(14)
where the sum with chosen weights \( w_j \) has been added to achieve a reduction of slopes and curvatures of the relaxation spectrum, as a possible further smoothing device. Applying the minimization conditions \( \partial \Phi / \partial a_i = 0 \) (\( i = 1, \ldots, 10 \)), one obtains a system of ten linear equations,

\[
\sum_{j=1}^{10} A_{ij} a_j = B_i \quad (i = 1, \ldots, 10)
\]

in which

\[
A_{ij} = \sum_{\alpha} \sum_{r} H_{i \alpha} H_{j \alpha} + \delta_{ij}, \quad B_i = \sum_{\alpha} \sum_{r} \left( E_{R \alpha} - \sum_{k=1}^{4} b_k x_k^{k-1} \right) H_{i \alpha},
\]

\( \delta_{ij} \) = Kronecker delta.

The computation may now proceed as follows. First one fits the \( E_{R \alpha N} \) values for all \( t'_{\alpha} \) by the polynomial (8). According to the method of least squares, coefficients \( b_1, \ldots, b_4 \) of this polynomial are solved from the equations

\[
\sum_{k=1}^{4} A'_{k\alpha} b_{k\alpha} = B'_{\alpha} \quad (k = 1, \ldots, 4), \text{ with } A'_{k\alpha} = \sum_{\alpha} x_{\alpha}^{k+2-\alpha}, \quad B'_{\alpha} = \sum_{\alpha} E_{R \alpha N} x_{\alpha}^{\alpha-1}.
\]

Subsequently coefficients \( A_{ij} \) and \( B_i \) are evaluated and parameters \( a_i \) are solved from (15). To check how close is the approximation of given data \( E_{R \alpha N} \), expression (12) may be evaluated.

A FORTRAN IV subroutine which implements the above algorithm, named MAXWL2, is listed in the Appendix. Comments within the program should be sufficient for easy use. After the polynomials for \( E_{\mu}(t') \) are obtained, one should integrate differential equations (1) to obtain the creep curves which correspond to \( E_{\mu}(t') \), and compare them with the original creep curves, \( J(t,t') \). This integration cannot be done by standard step-by-step algorithms because an increase of the time step with time would cause numerical instability. However, a general algorithm for efficient integration of structural creep problems with a creep law in the form of Eq. (1) has been presented in Ref. (5). Subroutine CRCURV applying this algorithm to the present problem is listed in the Appendix. For the underlying mathematics see Eqs. 15–19 from (5).

**Choice of Method**

Method 1 does not impose any specific analytical form on the spectrum and is thus suitable for obtaining little distorted shapes of relaxation spectra. Method 2 smoothes the spectra by a polynomial which gives more distorted shapes of the spectra but leads to fewer material parameters and
lessens the dependency of the spectra on small random components of given creep data. Either case may be preferable, depending on the nature of application. The cost of computing the spectra is minimal (about $1 on CDC 6400, at current rates).

**Numerical Examples**

Figs. 2 and 3 show by solid lines smoothed creep data of L'Hermite and Mamillan (12), which are compactly presented in Fig. 8, Ref. (5). The relaxation spectra computed from these data by Method 1 (MAXWL1, with weights $w_1 = .01, w_2 = .08$) appear in Fig. 4 and those computed by Method 2 (MAXWL2, with weights $w_1$ given in the comments in the program) appear in Fig. 5. The creep curves integrated back from the relaxation spectra in Figs. 4 and 5 are indicated in Figs. 2 and 3 by dashed lines. The fits are deemed to be satisfactory. Closer fits can be obtained, of course, by choosing a denser distribution of $\tau_\mu$-values and/or higher order polynomials. On the other hand, even a fit obtained by Method 2 with a quadratic polynomial (which may be computed by MAXWL2 if one assigns to $w_7, w_8, w_9, w_{10}$ a very large value, $10^{30}$) is acceptable (see dotted lines in Fig. 3).
Time Range Limitation of Spectra at Various Ages

Moduli $E_\mu(t')$ for times $\tau_\mu$ which are greater than about $30t'$ are irrelevant for the relaxation spectrum at age $t'$, because at that age the loading duration cannot exceed $t'$, so that no appreciable deformation rate can occur in the corresponding dashpots at age $t'$. Therefore, these moduli $E_\mu$ may be all lumped into one spring together with $E_n(t')$, which has been done in Figs. 4 and 5. (This lumping, however, need not be actually implemented in Eq. 6, in order to simplify programming.) The physically meaningful relaxation spectrum at age $t'$ ends at $\log \tau_\mu \approx \log (30t')$ and the $E_\mu$-values for higher $\log \tau_\mu$ are meaningless; these portions of spectra are not shown in Figures 4 and 5.

It is also noteworthy that all $E_\mu$-values are positive at all times, which is a condition that must be satisfied because of thermodynamic restrictions.

Acknowledgment

Support by the U.S. National Science Foundation under Grant No. GK-26030 is gratefully acknowledged.

References


SUBROUTINE RELAX (NDEC, JOEC, NA, JA, DOEF, T, TP, TPL, NT, ER)
C COMPUTES STRESS RESPONSE TO A GIVEN STRAIN HISTORY, AND RELAXATION
C FUNCTION IN PARTICULAR.
C INPUT- NDEC=NO. OF DECADES OF TIME IN LOG. SCALE* JDEC=NO. OF STEPS
C PER DECADE, NA=NO. OF AGES AT LOADING, JA=NO. OF AGES PER LOG(IO),
C T(I)=FIRST DISCRETE TIME, TP(I)=FIRST AGE AT LOADING, DOEF(I)=PRES-
C CRIBED STRAIN INCREMENT PER STEP FROM IT-1 TO IT, MINUS PRESCRIBED
C SHRINKAGE INCREMENT PER STEP, FOR RELAXATION ODEF(I)=1., ALL OTHER
C ODEF(I)=0. (IF SHRINKAGE IS CONSIDERED, INSERT DEPP=DEPP'0SHR(IT)
C AFTER STATEMENT NO. 00, OSHR(I)=SUPPLIED SHRINKAGE INCREMENTS.)
C SUBROUTINE CREEP FOR COMPUTING VALUES OF CREEP FUNCTION FROM T AND TP
C MUST BE SUPPLIED (SEE TABLE 2 OF REF.9),
C OUTPUT- ER(IT, I)=DISCRETE VALUES OF STRESS RELAXATION FUNCTION
C T(IT) AND TP(I): DISCRETE TIMES FROM LOADING AND DISCRETE AGES AT
C LOADING, NT=NO. OF DISCRETE TIMES.
C DIMENSION MUST BE AT LEAST OSTR(NT), ODEF(NT), T(NT), TP(NA),
C TPL(I), ER(NT, NA)
DIMENSION OSTR(30), ODEF(30), T(30), TP(30), TPL(30), ER(30, 8)
XX:1.0 FLOAT(JA) I
DO 11 IA = 1, NA
    TP(IA) = TP(IA-1)+XX
    TPL(I) = ALOG10(TP(IA))
    TPL(I+1) = ALOG10(TP(I+1))
    DTR(I) = IO.0*FLOAT(JDEC)
    NT = NDEC*JDEC
    DO 9 IT = 1, NT
        TI = (T(IT)-DTR(1))*2
        PRINT 16, (TP(IA): IA = I, NA)
16 FORMAT(1X,10.2, 2x, /, 4x, IOE, 2, 4x)
DO 50 IA = 1, NA
C COMPUTE THE FIRST STRESS VALUE USING EFFECTIVE MODULUS.
    ODEF(I)=ODEF(I)/CREEP(T(I), TP(I))
    ER(I, IA) = ODEF(I)
    DO 40 I = IT-1, 2
        I = I-1
        DEPP = 0.
        DO 30 JT = I + 1, IT
            DEPP = DEPP + ODEF(JT)*CREEP(IT, TP(JT)), CREEP(IT, TP(I))
        30 TP(I) = TP(I) + CREEP(IT, TP(I))
    20 ODEF(I) = ODEF(I) - DEPP
    10 CONTINUE
    PRINT 52
52 FORMAT('FORMAT(10.4, 1x, I4, 1x, I4) \n      ER(I, 1) ER(I, 2) ...
    DO 53 IT = 1, NT
        PRINT 54, IT, TP(I), ER(I, 1), IA = I, NA
54 FORMAT(1X,12.6, 3X, 10.12, 6/ 1X,10.12, 6/)
    RETURN
END
DO 202 NU=1,NMU
202 AI(NU,MU)=AI(NU,MU)+W
MU=1
202 AI(NU,MU)=AI(NU,MU)-W2
C SOLVE THE EQUATIONS BY A LIBRARY SUBROUTINE (NJ= AUXILIARY ARRAY)
CALL SOLSP(A,NMU,1I11,N11,11,11,N11,11,NJ)
DO 165 NU=1,NMU
165 AI(NU,IA)=AI(NU,IA)-W1-~2
C SOLVE THE EQUATIONS BY A LIBRARY SUBROUTINE (NJ= AUXILIARY ARRAY)
CALL SOLSP(A,B,NMU,1I11,N11,11,11,11,11)
10 CONTINUE
C SOLVE THE EQUATIONS BY A LIBRARY SUBROUTINE (NJ= AUXILIARY ARRAY)
CALL SOLSP(A,B,NMU,1I11,N11,11,11,11,11)
C FUNCTION MU OF EMU ASSUMED (FOR MU=I,...,MMU) AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) FOR EMU
C IN LOG(AGE TP AT LOADING) FOR EMU
C IN LOG(AGE TP AT LOADING) FOR EMU
IF(NA .LT. NMU) STOP
PRINT 806 FORMAT(WHICOF CUBIC POLYNOMIAL FOR EMU(NT,IA)=WEI2.3)
C NOW COMPUTE COEFFICIENTS AAA(J) OF CUBIC POLYNOMIAL IN MU AND LOG(AGE TP(IA) AT LOADING)
DO 580 IT=1,NT
580 SUM=SUM*PHI(MU)*FLOAT(MU+MUEX(1))*TP(IA)**(IXEX(1))
580 M(I,IA,IT)=SUM
C NOW COMPUTE EQUATION COEFFICIENT MATRICES A,B
DO 680 J=1,10
680 SUM=0.
DO 690 IA=1,NA
690 SUM=SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM
A2(J,L)=A2(J,L)**(I+J)**(I+J+2)
490 J=1,10
C SOLVE THE EQUATIONS BY A LIBRARY SUBROUTINE (NJ= AUXILIARY ARRAY)
CALL SOLSP(A,B,NMU,1I11,N11,11,11,11,11)
PRINT 506 FORMAT(WHICOF CUBIC POLYNOMIAL IN LOG(AGE) FOR EMU)
DO 510 IA=1,NU
510 SUM=SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM
B2(I,IA)=B2(I,IA)**(I+J)**(I+J+2)
506 DO 490 IT=1,NT
490 SUM=SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM
A2(I,IA)=A2(I,IA)**(I+J)**(I+J+2)
510 PRINT 506 FORMAT(WHICOF CUBIC POLYNOMIAL IN LOG(AGE) FOR EMU)
DO 510 IA=1,NU
510 SUM=SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM+SUM
B2(I,IA)=B2(I,IA)**(I+J)**(I+J+2)
510 CONTINUE
C COMPUTES RELAXATION MODULI EMU ASSUMED (FOR MU=I,...,MMU) AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) WITH COEFF. AAA, AND EMU FOR MU=NMU AS ONE-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TP) WITH COEFF. CE.
C ASSUMED EMU AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) WITH COEFF. AAA, AND EMU FOR MU=NMU AS ONE-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TP) WITH COEFF. CE.
C ASSUMED EMU AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) WITH COEFF. AAA, AND EMU FOR MU=NMU AS ONE-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TP) WITH COEFF. CE.
C ASSUMED EMU AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) WITH COEFF. AAA, AND EMU FOR MU=NMU AS ONE-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TP) WITH COEFF. CE.
C ASSUMED EMU AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) WITH COEFF. AAA, AND EMU FOR MU=NMU AS ONE-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TP) WITH COEFF. CE.
C ASSUMED EMU AS A TWO-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TIME T FROM LOADING) AND LOG(AGE TP AT LOADING) WITH COEFF. AAA, AND EMU FOR MU=NMU AS ONE-DIMENSIONAL CUBIC POLYNOMIAL IN LOG(TP) WITH COEFF. CE.
SUBROUTINE CURVIT(T,T,P,TAU,N,NA,MM,N,MMU,IALT,C,CE,AAA,MUEX,11)
C COMPUTES DISCRETE VALUES AJ(1,IA) OF CREEP FUNCTION J FROM GIVEN
C RELAXATION MODULI OF MAXWELL CHAIN.
C INPUT- T,T,T,P,TAU,N,NA,TAU,MM,NMU FROM SUBROUTINES RELAX
C AND MAXMXL OR MAXMXL (SEE THERE FOR DEFINITIONS). USER MUST SUPPLY
C SUBROUTINE FUNCTION EMU THAT COMPUTES EMU-VALUES FOR ANY AGE TP AND
C ANY MU. EMU(MU) IS COMPUTED FROM EITHER C(maxmll, IALT=1) OR
C C(CE,AAA,MUEX,11) (MAXWLL, IALT=2), IALT ON INPUT SELECTS THE OPTION.
C ARGUMENTS OF EMU(MU) APPEAR BELOW.
DIMENSION T(30),TT(30),TP(30),TAU(30),EMU(IO,10),A(I,10,11,10),C(10)
1 CE(10),EMUX(IO),CEX(IO),AAA(IO,10),MUEX(10)
C LOOP ON CREEP CURVES FOR VARIOUS AGES TP(IA) AT LOADING
DO 780 IA=I,NA
DEF=0.
DO 710 IA=I,NA
DO 730 IT=I,NT
SUM=SUM+(ER(MU,IA)-ERMA(IA))*(J,IA,IT)
610 SUM=SUM+(ER(I,IA)-ERMA(I,IA))*(J,IA,IT)
AAA(I,J)=SUM
DO 650 J=I,10
SUM=C.
IFI,J.EQ.0, I; SUM=0.
DO 640 IA=I,NA
DO 630 IT=I,NT
640 SUM=SUM+(I,IA,IT)*(J,IA,IT)
650 A3(I,J)=SUM
C SOLVE THE EQUATIONS BY A LIBRARY SUBROUTINE (NL=AUXILIARY ARRAY)
CALL SOXSP(A3,AAA,XO,X,I.E-6,IERR,iO,~L)
PRINT 751, AAA
751 FORMAT(5HOCOEFFICIENTS AAAII| OF CUBIC POLYNOMIAL FOR EMU(MU,IA)
IN MU AND LOGITP)/XXluE12.3/)
PRINT
761
761 FORMAT(5HVALUES OF EMU(MU,IA) COMPUTED FROM CUBIC POLYNOMIAL IN
1MU AND LOG(TP))
DO 770 IA=I,NA
00 766 MU=I,MMU
SUM=0.
DO 765 I=1,10
SUM=SUM+AAA(I)*(FLOAT(MU ~MUEX(I)|~TpLKIAI~*IXEX(I)|
765 AG(MU)=SUM
AG(MU)=SUM
SUM=30.*TP(IA)
DO 767 MU=I,MMU
IF(TAU(MU) .LT. SUM) 767
AG(MU)=AG(MU)/AG(MU)
AG(MU)=0.
767 CONTINUE
PRINT 768, IA~TP(IA)~ (AG(MUI, MU:I,NMU)
768 FORMAT(WH IA=I2,WH TP=FXC.3,GH EMU= BE11.3)
C AS A CHECK, COMPUTE CREEP FUNCTION VALUES FROM COEFF. AAA(I),CE(I)
CALL CRCURV(T,T,T,P,TAU,N,NA,MM,N,MMU,IALT,C,CE,AAA,MUEX,11)
RETURN
END
SUBROUTINE CRCURV(T,T,P,TAU,N,NA,MM,MMU,IALT,C,CE,AAA,MUEX,11)
C COMPUTES DISCRETE VALUES AJ(1,IA) OF CREEP FUNCTION J FROM GIVEN
C RELAXATION MODULI OF MAXWELL CHAIN.
C INPUT- T,T,T,P,TAU,N,NA,TAU,MM,NMU FROM SUBROUTINES RELAX
C AND MAXMXL OR MAXMXL (SEE THERE FOR DEFINITIONS). USER MUST SUPPLY
C SUBROUTINE FUNCTION EMU THAT COMPUTES EMU-VALUES FOR ANY AGE TP AND
C ANY MU. EMU(MU) IS COMPUTED FROM EITHER C(maxmll, IALT=1) OR
C C(CE,AAA,MUEX,11) (MAXWLL, IALT=2), IALT ON INPUT SELECTS THE OPTION.
C ARGUMENTS OF EMU(MU) APPEAR BELOW.
DIMENSION T(30),TT(30),TP(30),TAU(30),EMU(IO,10),A(I,10,11,10),C(10)
1 CE(10),EMUX(IO),CEX(IO),AAA(IO,10),MUEX(10)
C LOOP ON CREEP CURVES FOR VARIOUS AGES TP(IA) AT LOADING
DO 780 IA=I,NA
DEF=0.
DO 710 IA=I,NA
DO 730 IT=I,NT
SUM=SUM+(ER(MU,IA)-ERMA(IA))*(J,IA,IT)
610 SUM=SUM+(ER(I,IA)-ERMA(I,IA))*(J,IA,IT)
AAA(I,J)=SUM
DO 650 J=I,10
SUM=C.
IFI,J.EQ.0, I; SUM=0.
DO 640 IA=I,NA
DO 630 IT=I,NT
640 SUM=SUM+(I,IA,IT)*(J,IA,IT)
650 A3(I,J)=SUM
C SOLVE THE EQUATIONS BY A LIBRARY SUBROUTINE (NL=AUXILIARY ARRAY)
CALL SOXSP(A3,AAA,XO,X,I.E-6,IERR,iO,~L)
PRINT 751, AAA
751 FORMAT(5HOCOEFFICIENTS AAAII| OF CUBIC POLYNOMIAL FOR EMU(MU,IA)
IN MU AND LOGITP)/XXluE12.3/)
PRINT
761
761 FORMAT(5HVALUES OF EMU(MU,IA) COMPUTED FROM CUBIC POLYNOMIAL IN
1MU AND LOG(TP))
DO 770 IA=I,NA
00 766 MU=I,MMU
SUM=0.
DO 765 I=1,10
SUM=SUM+AAA(I)*(FLOAT(MU ~MUEX(I)|~TpLKIAI~*IXEX(I)|
765 AG(MU)=SUM
AG(MU)=SUM
SUM=30.*TP(IA)
DO 767 MU=I,MMU
IF(TAU(MU) .LT. SUM) 767
AG(MU)=AG(MU)/AG(MU)
AG(MU)=0.
767 CONTINUE
PRINT 768, IA~TP(IA)~ (AG(MUI, MU:I,NMU)
768 FORMAT(WH IA=I2,WH TP=FXC.3,GH EMU= BE11.3)
C AS A CHECK, COMPUTE CREEP FUNCTION VALUES FROM COEFF. AAA(I),CE(I)
CALL CRCURV(T,T,T,P,TAU,N,NA,MM,MMU,IALT,C,CE,AAA,MUEX,11)
RETURN
END