ELSEVIER

Computer Physics Communications 114 (1998) 236–242

Computer Physics Communications

TIMEDEL: A program for the detection and parameterization of resonances using the time-delay matrix

Darian T. Stibbe[1], Jonathan Tennyson

Department of Physics and Astronomy, University College London, Gower Street, London WC1E 6BT, UK

Received 25 March 1998

Abstract

TIMEDEL takes K-matrices from a scattering calculation, either read from a file or calculated on a dynamically adjusted grid, and calculates the time-delay matrix. This is then diagonalized to find the longest time-delay experienced by the scattering particle. The branching ratio of decay into the open channels is also found and optionally output. A resonance shows up as a characteristic Lorentzian form in the time-delay and the program searches the time-delay for maxima. It determines, from the overlap of adjacent resonances, whether each resonance should be fitted on its own or jointly with one or more other resonances. It performs the fitting and outputs the positions and widths of the resonances. © 1998 Elsevier Science B.V.

PACS: 34.80; 11.55; 03.80
Keywords: Resonances; Scattering; Time-delay matrix; K-matrix; Branching ratios

PROGRAM SUMMARY

Title of program: TIMEDEL

Catalogue identifier: ADJD

Program Summary URL: http://www.cpc.cs.qub.ac.uk/cpc/summaries/ADJD

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland

Licensing provisions: none

Computer for which the program is designed and others on which it is operable: Any computer with F90
Computers: IBM RS6000, DEC Alpha; *Installations:* Local workstation, 'Columbus' at RAL

Operating systems under which the program has been tested: AIX 4, DEC UNIX

Programming language used: Fortran 90

Memory required to execute with typical data: 5 Mwords

No. of bits in a word: 32

No. of bytes in distributed program, including test data, etc.: 964100

Distribution format: ASCII

Keywords: resonances, scattering, time-delay matrix, K-matrix, branching ratios

Nature of physical problem
TIMEDEL detects and parameterizes resonances found in scattering problems.

[1] E-mail: d.stibbe@ucl.ac.uk

Method of solution
K-matrices are read in or calculated on the fly (preferred option). The time-delay [1] is calculated from the K-matrices. The resonances are detected by examining the time-delay experienced by the incident particle and fitted to a characteristic Lorentzian or sum-of-Lorentzian form [2].

Restrictions on the complexity of the problem
The program will only find resonances which have definite maxima in the time-delay. Thus two resonances, very close in energy and width, will be fitted as one resonance. The program also assumes that the effect on the time-delay of more than one resonance is additive. Although this is normally the case, experience has shown this is not always true.

Typical running time
2.0 s plus the time taken by the user-supplied subroutine to calculate the K-matrices.

Unusual features of the program
The program uses a processor-dependent 'double complex' variable definition extension to the F90 standard. TIMEDEL can be used either with a user-supplied file of K-matrices or requires a subroutine (called here GETKMAT) supplied by the user to find the K-matrix of their problem at a particular energy. TIMEDEL also uses NAG library routines: E04FDF, a minimization routine; F04ADF, which solves the matrix equation **AS** = **B**; and F02HAF, which finds eigenvectors and values of a matrix. These routines are not supplied here and the code must be compiled with the NAG library [3].

References
[1] F.T. Smith, Phys. Rev. 114 (1960) 349.
[2] D.T. Stibbe, J. Tennyson, J. Phys. B 29 (1996) 4267.
[3] NAG Fortran Library Manual Mark 17, Numerical Algorithms Group (Oxford, UK, 1996).

LONG WRITE-UP

1. Introduction

When a particle collides with a target, it can, under certain conditions, be captured for a short time close to the target leading to the formation of a temporary compound state or resonance. Resonances can have a major effect on the cross section of various scattering processes, particularly those involving excitation of the target. In the case of scattering from a molecule, they can increase the rate of dissociation of the molecule by orders of magnitude [1].

The most common method of finding resonances has been to look at the K-matrix or eigenphase sum using standard programs such as RESFIT [2] and RESON [3]. A resonance will show up as a characteristic Breit–Wigner [4] form in the eigenphase sum ($\Delta(E)$) which can be fitted to find a position and width,

$$\Delta(E) = \Delta_0(E) + \tan^{-1}\left(\frac{\Gamma}{2(E_0 - E)}\right), \qquad (1)$$

where Γ is the (total) width of the resonance and E_0 is its position. $\Delta_0(E)$ is the sum of the background phases which is usually assumed to be a low-order polynomial. However, this fitting can be thwarted by a strongly varying background or the presence of more than one resonance.

There have been several more recently developed methods for fitting resonances. Quigley and Berrington [5,6] used the properties of the R-matrix in a procedure to fit automatically single or multiple resonances. Busby et al. [7,8] have written a graphical user interface to zoom interactively in on possible resonances which are then fitted using traditional methods. Spence and Scott [9] have borrowed the 'Hough transformation' from image processing to help detect resonances and Noble et al. [10] have attempted to find resonances through a direct search of the scattering matrix **S** within the complex energy plane.

In the time-delay method for fitting resonances [11], the time-delay matrix [12,13] is used to find the extra time of flight of the projectile due to its interaction with the target. A resonance shows up as a characteristic Lorentzian form in the longest time-delay, which can be fitted to find the relevant parameters. The resonances dominate the longest time-delay so greatly that the background can be treated as constant. This is particularly important for collisions involving molecules which have many degenerate channels, all of which contribute to the eigenphase sum. The fact that the background does not change is a great advantage over the eigenphase sum method.

If there is more than one resonance close by, the effect is usually additive so that the longest time delay, $q(E)$ in Eq. (3), appears as a sum of Lorentzians. The purpose of TIMEDEL is to calculate the time-delay as a function of energy from K-matrices, which are found routinely by most scattering calculations, and then to fit the resultant time-delay to Lorentzians to find the resonances. The program necessarily avoids thresholds by splitting the user-specified energy range into subranges bounded by the input threshold positions $\pm\epsilon$.

2. Method

The time-delay is found using the formulation of Smith [13]. The time-delay matrix **Q** is formed from the scattering matrix **S**, and the time operator $-i\,\mathrm{d}/\mathrm{d}E$,

$$\mathbf{Q} = -i\hbar \mathbf{S}^* \frac{\mathrm{d}\mathbf{S}}{\mathrm{d}E} . \tag{2}$$

Smith showed that the largest eigenvalue of the Q-matrix, q, represents the longest time-delay of the incident particle. He further showed that the probability of decay into a particular channel, the branching ratio β_i, is given by the square of the corresponding component of the eigenvector associated with q.

Close to resonance, the time-delay has a Lorentzian form given by

$$q(E) = \frac{\Gamma}{((E - E_0)^2 + (\Gamma/2)^2)} , \tag{3}$$

where E is the incoming projectile energy and E_0 is the position and Γ the width of the resonance. Exactly at resonance, the Lorentzian is at its maximum with, in atomic units, height = 4/width.

The computation can be split into two parts. The first part calculates the time-delay as a function of energy and the second locates and fits resonances.

2.1. *Calculating the time-delay*

The time-delay is calculated from K-matrices which can be supplied in one of two ways. Firstly, although not recommended, it is possible for TIMEDEL to read its input K-matrices from a file. As a derivative is to be calculated, the K-matrices must be provided as pairs with a small energy separation ΔE. The file should have the following form, repeated for the number of points at which the time delay is to be calculated (note that the file is read in free-format):

nopen energy dele
kmat1(1:nopen,1:nopen)
kmat2(1:nopen,1:nopen)

Nopen is the number of open channels, **energy** is the energy at which the time-delay is to be evaluated and **dele** is the separation in energy between the K-matrices **kmat1** and **kmat2**. **Kmat1** and **kmat2** are double precision K-matrices at energies **energy** − **dele**/2 and **energy** + **dele**/2. Sample data of this form is provided.

The second, preferred option is for the program to call a user-defined subroutine to find K-matrices at specific energies. In this case, the program can dynamically adapt the grid size depending on the magnitude of the time-delay. By making the spacing (**grid**) inversely proportional to the time-delay, there will always be approximately the same number of points for each resonance, independent of its width. This avoids the problem of fixed grids which often result in insufficient points to fit narrow resonances. The dynamic adaptation means that further calculations on finer and finer grids post priori are avoided, with a resultant saving in computer time. Furthermore, the ability to calculate K-matrices allows the branching ratios to be calculated at the exact resonance positions.

The user-supplied routine, GETKMAT, returns a K-matrix and the number of open channels at a given energy. A suggested structure to allow easy integration into existing codes is given in the appendix. This has been done for the UK molecular R-matrix suite of programs (see this issue [14]).

S-matrices are formed from the K-matrices by the subroutine KTOSMAT which uses NAG routine F04ADF [15] to solve the equation

$$(\mathbf{1} - i\mathbf{K})\mathbf{S} = (\mathbf{1} + i\mathbf{K}) . \tag{4}$$

For the time-delay at a particular energy E, it is necessary to find the S-matrices at $E - \Delta E/2$ and $E + \Delta E/2$. The value of the S-matrix at E is taken as the average of the above two matrices. $\mathrm{d}\mathbf{S}/\mathrm{d}E$ is approximated as $\Delta\mathbf{S}/\Delta E$ and the relevant values are then substituted into Eq. (2) to find the Q-matrix. The time-delay, $q_k(E)$, is the largest eigenvalue of the Q-matrix at point k, found using the NAG routine F02HAF [15]. The branching ratios for decay into each partial wave of all the open channels are the square of the elements of the eigenvector associated with the largest eigenvalue.

2.2. *Locating and fitting resonances*

Resonances are located approximately by finding maxima in the time-delay. In order to eliminate spurious maxima, five adjacent points (including the maximum point) are checked to be consistent with a maximum by looking at the gradients between points. An estimate of the width of each resonance is found from the maximum height. If the widths of two or more resonances are found to overlap sufficiently (determined by criterion input parameter **crit**), then they are put into a set of resonances to be fitted together.

The region of the time-delay to be fitted is determined by the input parameter **nes** or 'number each side'. This is the number of points each side of the maximum of the resonance (or maxima of the outermost resonances to be fitted together) which are to be used in the fitting procedure.

The fitting subroutine uses the NAG routine E04FDF [15] to fit the following functional form to the time-delay:

$$q(E) = \sum_{i=1,N} \frac{\Gamma_i}{((E - E_{0_i})^2 + (\Gamma_i/2)^2)} + bg, \qquad (5)$$

where N is the number of resonances to be fitted together and the E_{0_i}, Γ_i are the resonance parameters. The functional form includes a constant background (bg) which takes into account the effects of far away resonances. For each set (which contains either a single resonance or multiple resonances to be fitted together), the position and widths of the resonances are output. Also output is a dimensionless standard deviation of the time-delay to the functional form defined as the square root of the sum of the squares of the residues, normalized by the number of points and the maximum time-delay in the region under consideration,

$$\sigma = \frac{\sqrt{\sum_{k=1}^{M} (q_k - q(E))^2}}{M q_{\max}}. \qquad (6)$$

A 'good' fit should have a standard deviation smaller than around 10^{-2}.

In cases of numerical instability (for instance at the maximum of very narrow resonances) or for other reasons, it it possible that the time-delay will not vary smoothly. In this case, it is likely that the fitting routine will pick up spurious resonances or be thrown in its attempt to fit a real one. For this reason, or if a large standard deviation is observed, a useful check is to plot the time-delay on a linear-log scale and to look for resonances by eye. In the case of a very narrow resonance, the position can be found from the position of the maximum value of the time-delay and the width (in atomic units) is simply 4/maximum.

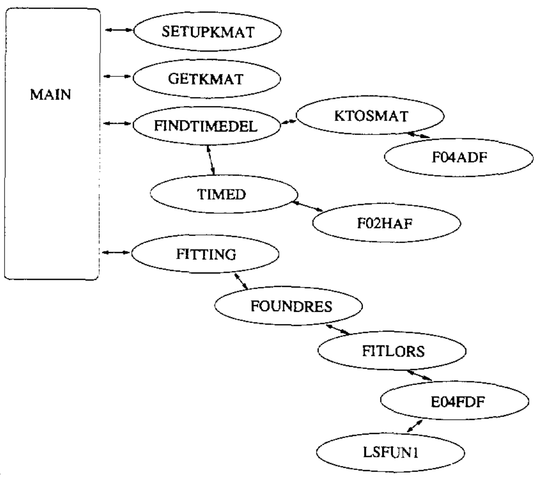

Fig. 1. Structure of the program. Note that the routines SETUPKMAT and GETKMAT are supplied by the user.

3. Program organization and data input

3.1. Organization

SETUPKMAT is a user-defined routine which sets up the calculation of K-matrices (only used if **adapt** is .true.).

GETKMAT is the user-defined routine which calculates a K-matrix at a required energy (only used if **adapt** is .true.).

FINDTIMEDEL takes a pair of K-matrices, converts them into S-matrices with the routine KTOSMAT (which uses NAG routine F04ADF [15]). These are then passed to TIMED which calculates the Q-matrix and finds its eigenvectors and eigenvalues using the NAG routine F02HAF [15]. The largest eigenvalue is the time-delay.

FITTING searches for maxima in the time-delay. If it finds any, it calls FOUNDRES which works out the sets of resonances that need to be fitted together. The fitting is performed in FITLORS using the NAG minimalization routine E04FDF [15], which itself requires routine LSFUN1 to find the error to a guessed functional form.

The error flag **ifail** from the NAG routine is output (along with other information output directly from the NAG routine). Only values of **ifail** of 1, 2, 4 and 8 are likely to produce incorrect results. The error trapping within the NAG routine is very sensitive and the message **ifail=5** will often be encountered. This can safely be ignored.

3.2. Input data

Input data [with defaults in parenthesis] is read from standard in via a namelist /TIME/. Further input might be required by the user's own K-matrix generator, if appropriate. The data type, with the exception of the logical variables, is given by the standard implicit double precision (a–h,o–z). When a file of K-matrices is being used, the default values will often suffice for the fitting. Note that in the distributed version of the program, even if only default values are to be used, an input namelist of the form '`&TIME; /`' is still required.

adapt [.false.] If .true., uses the adaptive grid (requires user-defined K-matrix generation program). If .false., inputs K-matrices from unit LUKMT.
crit [1.0]: Overlap criterion for resonances. The condition for fitting resonances 1 and 2 together is that $(E_1 - E_2) <$ **crit** $* (\Gamma_1 + \Gamma_2)$, where Γ is the resonance width.
ieunit [2]: Units used for all input energies: 0 = Hartrees, 1 = Rydbergs, 2 = eV.
einit : First energy of the region under consideration.
efinal : Final energy of the region under consideration.
epsil [0.01]: Time-delay is found to within energy **epsil** of the threshold positions.
gridinit [0.1]: Initial grid spacing.
gridmin [1.0D-8]: Minimum grid spacing allowed.
lubr [30] Logical unit for the output of branching ratios (if **writbr** is .true.).
lukmt [20] Logical unit for the input/output of user-generated K-matrices.
lutd [40]: Unit for the output of the time-delay.
delemax [0.00001]: Maximum allowed value of ΔE.
nes [10]: Number of points either side of a resonance maximum (or resonance maxima) to be fitted.
nptperres [50]: Approximate number of points per resonance to be calculated.
nthresh [0]: Number of thresholds to be considered (only used if **adapt** is .true.).
savek [.true.] If .true., writes any K-matrices generated to file **lukout**.
thresh(nthresh) : Array holding threshold energies if **nthresh** > 0 (only used if **adapt** is .true.).
writbr [.false.] If .true., then writes branching ratios to unit **lubr**.

4. Test case

The test data is a file of K-matrices produced from a calculation of the scattering of an electron from H_2^+ with total symmetry $^1\Sigma_g^+$ by Tennyson [16], based on an original calculation by Branchett and Tennyson [17]. For space reasons, only the K-matrices in the range $E = 18.094$ and 18.104 eV are provided, in which there are three resonances. The K-matrices were originally produced on an adaptive grid with initial grid size 1.0D–5 eV, and the number of points per resonance set at 30.

The time-delay (by default output as fort.40) is shown in Figs. 2a–c. The two resonances at energy around 18.1035 eV appear to be a single resonance at the low resolution of Fig. 2a but are clearly distinguished at the higher resolutions of Figs. 2b, c. In the latter figure, the adaptive grid can be seen clearly.

The test-case results, which used only default values for the input data, are given at the end of this paper.

Acknowledgements

This work was supported financially by CCP2 and the EPSRC. We are grateful to Hossein Sadeghpour for useful discussions.

Appendix A. Implementation of existing K-matrix routines

The code has been written to allow easy inclusion of user-defined K-matrix routines. In this discussion, it is assumed that the existing K-matrix generation code consists of initialization routines followed by a loop over energy in which the K-matrices are calculated. This is the case for the UK molecular R-matrix codes [14].

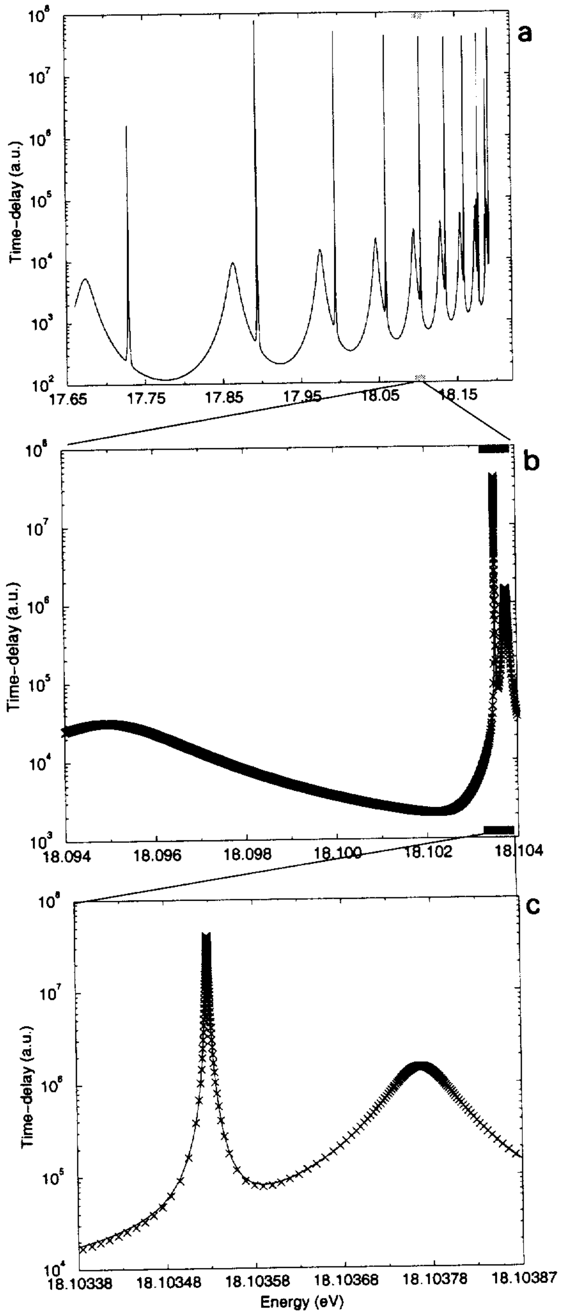

Fig. 2. (a) Time-delay against incoming electron energy for ${}^2\Sigma_g^+$ e-H_2^+ scattering. (b) The energy region used in the test case. (c) A further blow up of the two resonances on the right. The crosses are the calculated time-delay and the line is the fit to it.

The module USERDEFKMAT at the beginning of the program defines all the variables used in the K-matrix routines. The routine SETUPKMAT uses the module and performs the initialization before generation of K-matrices. The routine GETKMAT also uses the module (and hence shares all the variables with SETUPKMAT) and calculates a K-matrix for a given energy. It also returns the number of open channels.

References

[1] G.J. Schultz, Rev. Mod. Phys. 45 (1973) 423.
[2] K. Bartschat, P.G. Burke, Comput. Phys. Commun. 41 (1986) 75.
[3] J. Tennyson, C.J. Noble, Comput. Phys. Commun. 32 (1984) 421.
[4] G. Breit, E.P. Wigner, Phys. Rev. 49 (1936) 519.
[5] L. Quigley, K.A. Berrington, J. Phys. B 29 (1996) 4529.
[6] L. Quigley, K. Berrington, J. Pelan, Comput. Phys. Commun. 114 (1998) 225, this issue.
[7] D. Busby, N.S. Scott, P.G. Burke, C.J. Noble, Abstracts of the 23rd ATMOP Conference, Oxford, April 1–4 (1996).
[8] D.W. Busby, P.G. Burke, V.M. Burke, C.J. Noble, N.S. Scott, Comput. Phys. Commun. (1998) 114 (1998) 243, this issue.
[9] I.T. Spence, N.S. Scott, Abstracts of PECAM II, Belfast, July 23-26 (1996).
[10] C.J. Noble, M. Dörr, P.G. Burke, J. Phys. B 26 (1993) 2983.
[11] D.T. Stibbe, J. Tennyson, J. Phys. B 29 (1996) 4267.
[12] E.P. Wigner, Phys. Rev. 98 (1955) 145.
[13] F.T. Smith, Phys. Rev. 114 (1960) 349.
[14] L.A. Morgan, J. Tennyson, C.J. Gillan, Comput. Phys. Commun. 114 (1998) 120, this issue.
[15] NAG Fortran Library Manual Mark 17, Numerical Algorithms Group (Oxford, UK, 1996).
[16] J. Tennyson, At. Data & Nucl. Data Tables 64 (1996) 253.
[17] S.E. Branchett, J. Tennyson, J. Phys. B 25 (1992) 2017.

TEST RUN OUTPUT

```
Maxima found in this range

Attempting to fit resonance set  1 as 1 resonance(s)
 Output directly from NAG routine (if any) follows:
 ** It is probable that a local minimum has been found,
 ** but it cannot be guaranteed
 ** ABNORMAL EXIT from NAG Library routine E04FDF: IFAIL =     5
 ** NAG soft failure - control returned

 TIMEDEL Note: The above error message is due to
 over-sensitive error checking in the NAG routine.
 It can safely be ignored.

Fitted with background= .2129D+03 and St. Dev.= .1212D-07
Resonance  1. 1: Pos.= .18094918D+02 eV  Width= .35007588D-02 eV

Attempting to fit resonance set  2 as 1 resonance(s)
 Output directly from NAG routine (if any) follows:
 ** It is probable that a local minimum has been found,
 ** but it cannot be guaranteed
 ** ABNORMAL EXIT from NAG Library routine E04FDF: IFAIL =     5
 ** NAG soft failure - control returned

 TIMEDEL Note: The above error message is due to
 over-sensitive error checking in the NAG routine.
 It can safely be ignored.

Fitted with background= .4005D+02 and St. Dev.= .3638D-04
Resonance  2. 1: Pos.= .18103523D+02 eV  Width= .27097004D-05 eV

Attempting to fit resonance set  3 as 1 resonance(s)
 Output directly from NAG routine (if any) follows:
 ** It is probable that a local minimum has been found,
 ** but it cannot be guaranteed
 ** ABNORMAL EXIT from NAG Library routine E04FDF: IFAIL =     5
 ** NAG soft failure - control returned

 TIMEDEL Note: The above error message is due to
 over-sensitive error checking in the NAG routine.
 It can safely be ignored.

Fitted with background= .1370D+04 and St. Dev.= .1837D-05
Resonance  3. 1: Pos.= .18103762D+02 eV  Width= .73133478D-04 eV
```