technical memorandum

Daresbury Laborator

DL/SCI/TM69T

INTRODUCTION TO CHARGE-TRANSFER THEORY AND CALCULATIONS AND DOCUMENTATION FOR THE EIKONXS PROGRAM (PARALLEL VERSION)

by

R.J. ALLAN, SERC Daresbury Laboratory.

JULY, 1990

G90/344

Science and Engineering Research Council

DARESBURY LABORATORY

Daresbury, Warrington WA4 4AD

Introduction to charge–transfer theory and calculations and documentation for the EIKONXS program (parallel version).

R.J.Allan, Advanced Research Computing Group, S.E.R.C., Daresbury Laboratory, Warrington, WA4 4AD, U.K.

Abstract.

These notes are intended to document the EIKONXS program running on the Intel iPSC/2 multicomputer at Daresbury Laboratory in 1989. The program, derived from PAMPA, EIKON and TANGO (Gaussorgues, Piacentini and Salin [1,2]) calculates cross sections for total and angular differential charge transfer and electronic excitation, orbital alignment and orientation parameters and radiative transitions during the collisions of ions and atoms. The treatment uses adiabatic and diabatic molecular wavefunctions with a common electron translation factor and classical trajectory.

A formal derivation is given of some of the equations used in the program. This may serve as a concise introduction to the theory of molecular charge transfer.

Contents.

I Theory
II Introduction and general use of the program
III Program modules and useage
IV Listing the molecular data
References
Appendix A File structure, format and default names
Appendix B Makefile for Intel iPSC/2
Appendix C Optional partial linking of routines

I Theory.

I briefly outline here a novel derivation of the quasiclassical close–coupled channel equations with the form of common electron translation factor (ETF) first prescribed by Errea et al. [20]. Some comments on the physical significance of the equations are given during their derivation. The resulting finite set of first–order coupled linear differential equations in one variable is what will be solved by the program.

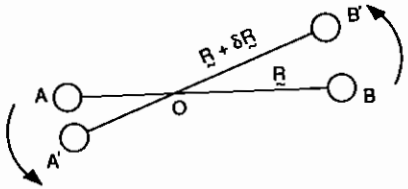
Consider the time–dependent Schroedinger equation

$$(H - i\hbar \frac{\partial}{\partial t}|_{SF}) \Psi = 0 \qquad (1)$$

In the quasiclassical method the nuclear coordinates are described solely by a parametric dependence of the quantities on internuclear vector R(t), thus $H \equiv H_{el}$ the Born–Oppenheimer electronic hamiltonian for fixed nuclei.

It is the $\partial/\partial t$ operator which causes transitions (excitation and charge transfer) between "stationary" molecular eigenstates. We can now interpret the physical significance with the aid of figure 1.

Fig. 1 Development of the collision system during time $t \rightarrow t+\delta t$



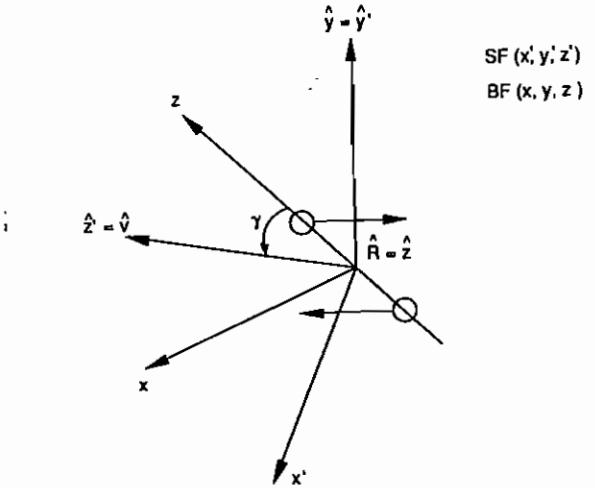
$|\chi|^2$

At time t the molecular wavefunction $\chi_k(R)$ is an eigenstate of the system AB with fixed nuclear configuration R(t); the zeroth order in our many–body expansion. The next step is to act with the $\partial/\partial t$ operator whilst keeping this electronic distribution fixed in space. In a time δt the nuclei move to positions given by $\mathbf{R}+\delta\mathbf{R}$. This operator modifies the electron distribution. The modified distribution must now be projected onto a linear combination of new eigenstates $\chi'_k(R) \rightarrow c^n_k \chi_n(R+\delta R)$ and it is this linear combination which follows the transitions during time $t \rightarrow t+\delta t$ for which the amplitudes are $c^n_k(R)$. The $\partial/\partial t$ operator is not simply a rotation and translation of the charge–cloud in space following the motion of the nuclei, otherwise the electronic evolution would follow an adiabatic path. The second term of the expansion is thus essentially from the

$\partial/\partial t$ operator. There are higher terms which have not yet been identified but for which we substitute the ETF.

The time-derivative operator $\partial/\partial t|_{SF}$ acts in a space-fixed (SF) reference frame whereas both H(R) and $\Psi(R)$ are normally represented in body-fixed (MF) coordinates. We therefore need to rotate SF into BF to obtain the consistent operator (see figure 2).

Fig.2 SF and BF coordinate systems for a diatomic molecule.



The effect of $\partial/\partial t|_{SF}$ acting on the MF coordinates (x,y,z) and the parameter R is

$$\frac{\partial}{\partial t}\Big|_{SF} = \frac{\partial x}{\partial t}\Big|_{SF}\frac{\partial}{\partial x}\Big|_{MF} + \frac{\partial y}{\partial t}\Big|_{SF}\frac{\partial}{\partial y}\Big|_{MF} + \frac{\partial z}{\partial t}\Big|_{SF}\frac{\partial}{\partial z}\Big|_{MF} + \frac{dR}{dt}\Big|_{SF}\frac{\partial}{\partial R}_{MF} \qquad (2)$$

For more complicated systems there may be other parameters (particularly for atom-molecule collisions for which this analysis still holds). If we suppose that the coordinates q are related by

$$q|_{SF} = \Re^{-1} q|_{MF},\ q|_{MF} = \Re q|_{SF} \qquad (3)$$

where $\Re(t)$ is a rotation matrix and $\Re^{-1}$ the reverse rotation, then

$$\frac{\partial}{\partial t}\Big|_{SF} q|_{MF} = \Re q|_{SF} + \Re v_q|_{SF}$$
$$= \Re\Re^{-1} q|_{MF} + v_q|_{MF} \qquad (4)$$

specifically for the diatomic case $\Re = \begin{bmatrix} S & 0 & -C \\ 0 & 1 & 0 \\ C & 0 & S \end{bmatrix}$ (5)

where S = sinγ, C=cosγ and γ is the rotation about the y axis which relates the two frames (see figure 2).

$$\text{therefore}\quad \dot{\Re}\Re^{-1} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \dot{\gamma} \qquad (6)$$

$$\text{which yields}\quad \begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -z \\ x \end{bmatrix} \dot{\gamma} \qquad (7)$$

since $v_q|_{SF}$ is zero for a charge cloud fixed in space during the interval δt. It is the electronic velocity field contribution represented by the second term of (4) which will be approximated by the ETF. This term is non-zero if the electrons are considered to be moving with the nuclei.

We furthermore allow a shift of origin along the molecular axis to a position PR from one of the atoms. This atom is referred to as the AOF reference frame

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix}_{AOF} = \begin{bmatrix} -z-PR \\ x \end{bmatrix} \dot{\gamma} - \begin{bmatrix} 0 \\ \dot{PR} \end{bmatrix} \qquad (8)$$

A similar expression holds for a centre on the other atom

$$\text{Finally}\quad \frac{\partial}{\partial t}\Big|_{SF} = \frac{i}{\hbar}L_y\dot{\gamma} - PR\dot{\gamma}\frac{\partial}{\partial x} - \dot{PR}\frac{\partial}{\partial z} + \dot{R}\frac{\partial}{\partial R}. \qquad (9)$$

This is the basic expression for the operator in the perturbed-stationary-states treatment of charge transfer and electronic excitation in diatomic systems.

We next consider the form adopted for the molecular expansion of the scattering wavefunction Ψ, and in particular the guage transformation which leads to the common translation factor.

$$\Psi \equiv c^n\varphi_n \equiv c^n\chi_n e^{iU(r,R)} \qquad (10)$$

We adopt covarient notation where a repeated index indicates summation (Einstein's convention). The guage transformation contains a phase factor U which we shall treat explicitly.

If the wavefunctions are orthonormal $\langle\varphi_m|\varphi_n\rangle=\delta^m_n$ then substituting in (1) and projecting onto $\langle\varphi_m|$ yields

$$i\hbar\frac{\partial c^m}{\partial t} = \langle\chi_m e^{iU}|H - i\hbar\frac{\partial}{\partial t}|\chi_n e^{iU}\rangle c^n \qquad (11)$$

Now consider that $H\chi_k \equiv \varepsilon_k\chi_k;\ \forall\ k$, where this defines the wavefunctions $\{\chi\}$ to be solutions of the electronic hamiltonian (stationary states), and noting that the hamiltonian contains a term $-\hbar^2\nabla^2_r/2m_e$ which acts on the various electronic

coordinates, of which U is a function. Setting $m_e=1$ and $\hbar=1$ then

$$H\varphi_k = (H\chi_k)e^{iU} - (\nabla\chi_k)\nabla e^{iU} - \chi_k\nabla^2 e^{iU}/2$$
$$= e^{iU}[\varepsilon_k - i(\nabla U)\nabla - i\nabla^2 U/2 + (\nabla U)^2/2]\chi_k \quad (12)$$

and $i\frac{\partial}{\partial t}\chi_k e^{iU} = e^{iU}\left[i\frac{\partial}{\partial t} - \dot{U}\right]\chi_k$ in (11)

thus writing $i\frac{\partial c_n^m}{\partial t} \equiv M_n^m c_n^n$

$$M_n^m = \langle\chi_m|\varepsilon_n - i\frac{\partial}{\partial t}|\chi_n\rangle + \langle\chi_m|(\nabla U)^2 + 2\dot{U}|\chi_n\rangle/2 - i\langle\chi_m|\nabla^2 U/2 + (\nabla U)\nabla|\chi_n\rangle \quad (13)$$

and we note that

$$[H,U] = -\nabla^2 U/2 - (\nabla U)\nabla \quad (14)$$

so that

$$M_n^m = \langle\chi_m|\varepsilon_n - i\frac{\partial}{\partial t}|\chi_n\rangle + \langle\chi_m|(\nabla U)^2 + 2\dot{U}|\chi_n\rangle/2 - i(\varepsilon_m - \varepsilon_n)\langle\chi_m|U|\chi_n\rangle \quad (15)$$

we now define U to depend on the switching function f

$$U \equiv f(r,R)\underline{v}.\underline{r} - f^2v^2t/2 \quad (16)$$

and in particular we choose the functional form of Errea et al. [20]

$$f \equiv G(R)z \text{ in the MF coordinates.} \quad (17)$$

Substituting first (16) into (15)

$$M_n^m = \langle\chi_m|\varepsilon_n - i\frac{\partial}{\partial t}|\chi_n\rangle + \langle\chi_m|(\underline{v}.\underline{r})^2(\nabla f)^2 + f^2(\nabla f)^2v^4t^2 + 2f(\underline{v}.\underline{r})\underline{v}.\nabla f$$
$$- 2f^2v^2t\underline{v}.\nabla f - 2(\underline{v}.\underline{r})v^2tf(\nabla f)^2 + 2\dot{f}\underline{v}.\underline{r} - 2f\dot{f}v^2t|\chi_n\rangle/2$$
$$- i(\varepsilon_m - \varepsilon_n)\langle\chi_m|f\underline{v}.\underline{r} - f^2v^2t/2|\chi_n\rangle \quad (18)$$

It is now necessary to look at the form of the relative velocity v using the rotation matrix (5) in a consistent manner

$$q|_{SF} = \Re^{-1}q|_{MF}$$

therefore $v_q|_{MF} = \Re\frac{\partial}{\partial t}|_{SF}\Re^{-1}q|_{MF} = \Re\dot{\Re}^{-1}q|_{MF} + \dot{q}|_{MF}$ (19)

In MF coordinates explicitly, for a straight-line trajectory,

$$R = \rho + vt \quad (20)$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}\dot{\gamma} + \frac{\partial}{\partial t}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} R\dot{\gamma} \\ 0 \\ \partial R/\partial t \end{bmatrix} \quad (21)$$

where $\hat{z}=\hat{R}$ in these coordinates

also $v^2t = v.v.t = v_zR$ independent of origin.

Using the expression for the operator (9) in the MF coordinates

$$\frac{\partial f}{\partial t} = \left[x\dot{\gamma}\frac{\partial}{\partial z} - z\dot{\gamma}\frac{\partial}{\partial x} + \frac{dR}{dt}\frac{\partial}{\partial R}\right]f$$

$$= Gx\dot{\gamma} + z\frac{dGdR}{dRdt} = G\frac{x}{R}v_x + Gzv_z\frac{dG}{dR} \quad (22)$$

and also note that

$$\underline{v}.\underline{r} = v_xx + v_zz \quad (23)$$

We can substitute into (18) to give

$$M_n^m = \langle\chi_m|\varepsilon_n - iv_z\frac{d}{zdR} + i\frac{v}{R}x\frac{i}{\hbar}L_y\ |\chi_n\rangle$$
$$+ \langle\chi_m|x^2|\chi_n\rangle v_x^2(\frac{G^2}{2} + \frac{G}{R})$$
$$+ \langle\chi_m|z^2|\chi_n\rangle v_z^2[\frac{3}{2}G^2 + \frac{1}{2}G^4R^2 - 2G^3R + \frac{dG}{dR} + GR\frac{dG}{dR}] + i(\varepsilon_m - \varepsilon_n)\frac{G}{v_z} - \frac{i}{2}(\varepsilon_m - \varepsilon_n)\frac{G^2R}{v_z}$$
$$+ \langle\chi_m|xz|\chi_n\rangle v_xv_z[G^2 - RG^3 + \frac{1}{R}G + \frac{dG}{dR} + i(\varepsilon_m - \varepsilon_n)\frac{G}{v_z}]. \quad (24)$$

Now we use the relations from the commutators

$$\langle\chi_m|z^2|\chi_n\rangle = -2\langle\chi_m|z\frac{\partial}{\partial z}|\chi_n\rangle/(\varepsilon_m - \varepsilon_n)$$
$$\langle\chi_m|xz|\chi_n\rangle = -\langle\chi_m|x\frac{\partial}{\partial z} + z\frac{\partial}{\partial x}|\chi_n\rangle/(\varepsilon_m - \varepsilon_n) \quad (25)$$

which gives

$$M_n^m = \langle\chi_m|\varepsilon_n - iv_z\frac{d}{zdR} + i\frac{v}{R}x\frac{i}{\hbar}L_y\ |\chi_n\rangle$$
$$-2i\langle\chi_m|z\frac{\partial}{\partial z}|\chi_n\rangle v_z(G - \frac{1}{2}G^2R)$$
$$-i\langle\chi_m|x\frac{\partial}{\partial z} + z\frac{\partial}{\partial x}|\chi_n\rangle v_xG$$
$$-2\langle\chi_m|z\frac{\partial}{\partial z}|\chi_n\rangle v_z^2[\frac{3}{2}G^2 + \frac{1}{2}G^4R^2 - 2G^3R + \frac{dG}{dR} + GR\frac{dG}{dR}]/(\varepsilon_m - \varepsilon_n)$$
$$-\langle\chi_m|x\frac{\partial}{\partial z} + z\frac{\partial}{\partial x}|\chi_n\rangle v_xv_z[G^2 - RG^3 + \frac{1}{R}G + \frac{dG}{dR}]/(\varepsilon_m - \varepsilon_n)$$
$$+ \langle\chi_m|x^2|\chi_n\rangle v_x^2(\frac{G^2}{2} + \frac{G}{R}). \quad (26)$$

Now look at the transformation to AOF coordinates. This involves replacing z by z+PR, which gives rise to the following *extra* terms in the above equation

$$\langle\chi_m|iPv_z\frac{\partial}{z\partial z} + iPv_x\frac{\partial}{x\partial x} - 2iPRv_z(G - \frac{1}{2}G^2R)\frac{\partial}{\partial z} - iv_xGPR\frac{\partial}{\partial x}|\chi_n\rangle$$
$$-2v_z^2PR\langle\chi_m|\frac{\partial}{\partial z}|\chi_n\rangle[\frac{3}{2}G^2 + \frac{1}{2}G^4R^2 - 2G^3R + \frac{dG}{dR}(1 - GR)]/(\varepsilon_m - \varepsilon_n)$$
$$-v_xv_zPR\langle\chi_m|\frac{\partial}{\partial x}|\chi_n\rangle[G^2 - RG^3 + \frac{1}{R}G + \frac{dG}{dR}]/(\varepsilon_m - \varepsilon_n). \quad (26a)$$

The physical purpose (one of them) of the ETF is to make the cross sections independent of the coordinate systems used for the calculation (Galilean invarience). One trial way to achieve this is to remove the dependence on P above. This will fix the form we need for G(R). Taking the v_x and v_z dependent terms separately gives simultaneous equations for unknown G:

$$\frac{3}{2}G^2+\frac{1}{2}G^4R^2-2G^3R+\frac{dG}{dR}(1-GR) = 0$$

$$G^2-RG^3+\frac{1}{R}G+\frac{dG}{dR} = 0 \qquad (27)$$

Eliminating the term in dG/dR and solving the resulting cubic

$$G = \frac{1}{R}, \frac{1}{R}, \frac{2}{R}$$

Errea *et al.* [20] chose the first of these, and we shall do likewise since a constant is not important. We note that remaining terms in (26a) also vanish with this choice, so finally

$$i\frac{\partial c^m}{\partial t} = \{<\chi_m|\varepsilon_n - i\frac{dRd}{dt\,dR} - i\frac{d\gamma i}{dt\,\hbar}L_y|\chi_n>$$
$$-i\frac{dR}{dt}<\chi_m|z\frac{\partial}{\partial z}|\chi_n>/R$$
$$-i\frac{d\gamma}{dt}<\chi_m|x\frac{\partial}{\partial z}+z\frac{\partial}{\partial x}|\chi_n> + 1.5(\frac{d\gamma}{dt})^2<\chi_m|x^2|\chi_n>\}c^n \qquad (28)$$

These are the coupled equations which are solved by the program to give the transition amplitudes {c}. We note that in fact we may replace c^k by a vector c^k and solve simultaneously for a linearly independent set of initial conditions; the operators are all the same and this will not involve much extra computation. In practice we solve a set of matrix equations with starting conditions $c_n^m(-\infty) = \delta_n^m$.

The data required by the program is as follows:

i) adiabatic molecular energies ε_n

ii) Radial coupling and translation factor term $<\chi_m|\frac{\partial}{\partial R}|\chi_n>$ and $<\chi_m|z\frac{\partial}{\partial z}|\chi_n>$

iii) Rotational coupling and translation factor term $<\chi_m|\frac{iL_y}{\hbar}|\chi_n>$ and $<\chi_m|x\frac{\partial}{\partial z}+z\frac{\partial}{\partial x}|\chi_n>$

iv) and the x^2 matrix element $<\chi_m|x^2|\chi_n>$.

These are stored in a file as described in Appendix A.

II Introduction and general use of the program.

The directory structure used in maintaining the program is ~/eikonxs for the source files described in section 3. The control files and molecular data are contained in a subdirectory ~/eikonxs/hehp for instance. The program (parallel) is run on the Intel machine by invoking

../eikonxs0.out > output.txt &

on the Convex by typing

../eikonxs.out < direct.con > output.txt &

where direct.con is the control file now described, or on the Alliant parallel complex of processors by typing

execute -c5../eikonxs.out < direct.con > output.txt &

for instance if 5 processors are required.

In general the control file read on LU5 (it is always called eikonxs0.con and an identical copy eikonxs.con in the parallel version, or an arbitrary name, in the shared-memory version) begins with a line telling the main program how many processors are to be used (a power of two) and the name of the job (format i2,a8). The program input data then consists of a set of directives in upper case, which pass control on to the various modules named above. These are described separately in the rest of this documentation. The directives are, in order, APROB, COPY, UNCOPY, CLASSICAL, DEFLN, DIAB, DIFFXS, DPROB, EMISSION, GAMMA, GAUCHO, GRAPHICS, ORIENT, RADIATION, STARK, and TOTAL. Further directives NOPR and TITLE control output listing on LU6 and LU4. Directive DOUBLE is useful if two different scattering symmetries (e.g. gerade and ungerade) have to be combined to obtain cross sections. Directives END, ADD and NONE are used in certain of the modules. NONE and END are interchangeable. Directive STOP is used to terminate execution.

Example of input on LU5 to control the entire application

```
16eikonxs
NOPR
TITLE
****** NA-H+ 7 ESTADOS, rmax=30.  nro=203 ** 7 etats couples
APROB
0.
001
0.1414
004 1.
```

```
080 0.076
050 0.1
045 0.15
027 0.22
-45.
45.
0.707
1.0426
45.
  0  0
-0.18848  -0.125    -0.125    -0.11051  -0.07159  -0.125
-0.11155
  0  0  0  0  0  1  1
F
0.0
TOTAL
999
 **** Transicion NA(3S) ——> H(2P) *****
  1  2  0.707
  1  3  -0.707
  0  0
COPY
STARK
NONE
-0.18848  -0.125    -0.125    -0.11051  -0.07159  -0.125
-0.11155
  0  2  2  5  0  3  6
TOTAL
999
****** transicion na(3s) ——> h(2s) *****
  1  2  0.707
  1  3  0.707
999
**** TRANSICION Na(3s) —> H(2p sig) ****
  1  2  0.707
  1  3  -0.707
999
**** TRANSICION Na(3s) —> H(2p pi) ****
  1  6  1.
999
**** TRANSICION Na(3p sig) —> H(2s) ****
  4  2  0.707
  4  3  0.707
999
**** TRANSICION Na(3p sig) —> H(2p sig) ****
  4  2  0.707
  4  3  -0.707
999
**** TRANSICION Na(3p sig) —> H(2p pi) ****
  4  6  1.
999
**** TRANSICION Na(3p pi) —> H(2s) ****
  7  2  0.707
  7  3  0.707
```



```
999
 **** TRANSICION Na(3p pi) —> H(2p sig) ****
  7  2  0.707
  7  3  -0.707
999
 **** TRANSICION Na(3p pi) —> H(2p pi) ****
  7  6  1.
  0  0
TITLE
**** SECCIONES DIFERENCIALES PARA 7 ESTADOS *********
TITLE
 Na-H+ 7 estados,
EIKON
 80
0.01
0.01
  0  2
1.
1.
0.
0.
1.0426
999
 right circularly polarised H(2p) from right circularly polarised Na(3p)
  4  2  0 0.35355
  4  3  0 -0.35355
  4  6  1          0.5
  7  2  1          0.35355
  7  3  1          -0.35355
  7  6  0 -0.5
999
right - left
  4  2  0 0.35355
  4  3  0 -0.35355
  4  6  1          -0.5
  7  2  1          0.35355
  7  3  1          -0.35355
  7  6  0 -0.5
999
left - right
  4  2  0 0.35355
  4  3  0 -0.35355
  4  6  1          0.5
  7  2  1          -0.35355
  7  3  1          0.35355
  7  6  0 0.5
999
left- left
  4  2  0 0.35355
  4  3  0 -0.35355
  4  6  1          -0.5
  7  2  1          -0.35355
  7  3  1          0.35355
  7  6  0 -0.5
```

```
999
right - 2s
  4  2  0 0.5
  4  3  0 0.5
  7  2  1          0.5
  7  3  1          0.5
999
left- 2s
  4  2  0 0.5
  4  3  0 0.5
  7  2  1          -0.5
  7  3  1          -0.5
  0  0
STOP
```

Example of output listing from above run on Alliant FX2800

```
***** EIKONXS ***** running on Alliant FX/2812 at Daresbury
****** NA-H+ 7 ESTADOS, rmax=30.  nro=203 ** 7 etats couples
*****PROGRAM.TANGO***** PROPAGATION ON TRAJECTORY
 alpha =    0.00000
reading nr, ns =  37  7
iperm   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
iperm  21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
particion   no. de b    incr. b   binic. = 1.00
            80       0.0760
            50       0.1000
            45       0.1500
            27       0.2200
 Velocidad =    0.28280    Energia =2082.91434  eV.

RIN= -45.000         RM=   45.000         HIN= 0.50000         AMASS=   1914.3
RPHAS=  45.000

TRANSITION PROBABILITIES

.....END OF RUN   1 proc   -1
*****PROGRAM.TOTAL*****

RELATIVE VELOCITY  0.28280    [AU]
TOTAL CROSS SECTIONS [CM~2*10~-16]
203 IMPACT PARAMETERS FROM   1.0000    TO  24.770    [ao]

INITIAL STATE   1 TO:
  1    850.27       2    37.720       3    7.9227       4    81.581       5    13.545       6
5.6069       7    63.720
 INITIAL STATE   2 TO:
  1    37.720       2    665.88       3    135.89       4    34.638       5    40.023       6
94.912       7    62.123
 INITIAL STATE   3 TO:
  1    7.9225       2    135.89       3    619.17       4    42.241       5    14.274       6
248.19       7    31.208
 INITIAL STATE   4 TO:
  1    81.581       2    34.639       3    42.241       4    561.30       5    62.310       6
```



```
32.194       7    215.90
 INITIAL STATE   5 TO:
  1    13.545       2    40.022       3    14.273       4    62.309       5    1176.7       6
46.763       7    77.397
 INITIAL STATE   6 TO:
  1    5.6070       2    94.912       3    248.19       4    32.195       5    46.763       6
631.26       7    52.144
 INITIAL STATE   7 TO:
  1    63.720       2    62.123       3    31.208       4    215.90       5    77.395       6
52.145       7    523.61

 **** Transicion NA(3S) ---> H(2P) *****
  1  2( 0.70700    ,  0.0000    )  1  3(-0.70700    ,  0.0000    )
 XS =   23.195
*****PROGRAM.COPY*****  2 TO   9
*****PROGRAM.STARKS***** LONG-RANGE EXTRAPOLATION
  1 203   7  45.000
 ME =   0  2  2  5  0  3  6
.....BLOCK   2 OF MATRIX AT ROW  2
.....BLOCK   3 OF MATRIX AT ROW  5
.....BLOCK   4 OF MATRIX AT ROW  7
*****PROGRAM.TOTAL*****

RELATIVE VELOCITY  0.28280    [AU]
TOTAL CROSS SECTIONS [CM~2*10~-16]
203 IMPACT PARAMETERS FROM   1.0000    TO  24.770    [ao]

INITIAL STATE   1 TO:
  1    850.52       2    36.423       3    7.4346       4    70.496       5    13.546       6
4.9116       7    52.593
 INITIAL STATE   2 TO:
  1    36.423       2    633.25       3    105.99       4    29.016       5    38.795       6
71.627       7    50.065
 INITIAL STATE   3 TO:
  1    7.4344       2    106.00       3    606.64       4    35.431       5    13.748       6
197.69       7    29.554
 INITIAL STATE   4 TO:
  1    70.496       2    29.017       3    35.432       4    559.52       5    52.609       6
24.949       7    140.61
 INITIAL STATE   5 TO:
  1    13.547       2    38.794       3    13.747       4    52.609       5    1179.9       6
42.663       7    65.994
 INITIAL STATE   6 TO:
  1    4.9117       2    71.626       3    197.69       4    24.950       5    42.663       6
604.96       7    43.381
 INITIAL STATE   7 TO:
  1    52.594       2    50.065       3    29.554       4    140.62       5    65.993       6
43.381       7    519.89

****** transicion na(3s) ---> h(2s) *****
  1  2( 0.70700    ,  0.0000    )  1  3( 0.70700    ,  0.0000    )
 XS =   22.438
**** TRANSICION Na(3s) --> H(2p sig) ****
  1  2( 0.70700    ,  0.0000    )  1  3(-0.70700    ,  0.0000    )
```

```
XS =  21.419
**** TRANSICION Na(3s) —> H(2p pi) ****
 1  6(  1.0000    ,  0.0000    )
XS =  4.9116
**** TRANSICION Na(3p sig) —> H(2s) ****
 4  2( 0.70700    ,  0.0000    ) 4  3( 0.70700    ,  0.0000    )
XS =  15.697
**** TRANSICION Na(3p sig) —> H(2p sig) ****
 4  2( 0.70700    ,  0.0000    ) 4  3(-0.70700    ,  0.0000    )
XS =  48.752
**** TRANSICION Na(3p sig) —> H(2p pi) ****
 4  6(  1.0000    ,  0.0000    )
XS =  24.949
**** TRANSICION Na(3p pi) —> H(2s) ****
 7  2( 0.70700    ,  0.0000    ) 7  3( 0.70700    ,  0.0000    )
XS =  60.678
**** TRANSICION Na(3p pi) —> H(2p sig) ****
 7  2( 0.70700    ,  0.0000    ) 7  3(-0.70700    ,  0.0000    )
XS =  18.940
**** TRANSICION Na(3p pi) —> H(2p pi) ****
 7  6(  1.0000    ,  0.0000    )
XS =  43.381
**** SECCIONES DIFERENCIALES PARA 7 ESTADOS **********
 Na-H+ 7 estados,
 right circularly polarised H(2p) from right circularly polarised Na(3p)
SPEED  0.28280    PROCESS  4 TO  2 WEIGHT (  0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  3 WEIGHT ( -0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  6 WEIGHT (  0.0000    ,  0.50000    )
SPEED  0.28280    PROCESS  7 TO  2 WEIGHT (  0.0000    ,  0.35355    )
SPEED  0.28280    PROCESS  7 TO  3 WEIGHT (  0.0000    , -0.35355    )
SPEED  0.28280    PROCESS  7 TO  6 WEIGHT ( -0.50000    ,  0.0000    )
END OF RUN on proc  -1
  energia (eV) =2082.91434
ang(grad)  ang x Ek    secc. diferenciales

0.10000E-01  20.829     1.9147     4.7159
0.20000E-01  41.658     0.69099    11.209
      ...
0.79000      1645.5     198.34     0.14852E-05
0.80000      1666.3     208.83     50.642
right - left
SPEED  0.28280    PROCESS  4 TO  2 WEIGHT (  0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  3 WEIGHT ( -0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  6 WEIGHT (  0.0000    , -0.50000    )
SPEED  0.28280    PROCESS  7 TO  2 WEIGHT (  0.0000    ,  0.35355    )
SPEED  0.28280    PROCESS  7 TO  3 WEIGHT (  0.0000    , -0.35355    )
SPEED  0.28280    PROCESS  7 TO  6 WEIGHT (  0.50000    ,  0.0000    )
END OF RUN on proc  -1
  energia (eV) =2082.91434
ang(grad)  ang x Ek    secc. diferenciales

0.10000E-01  20.829     2.8856     4.7161
0.20000E-01  41.658     2.9382     11.208
      ...
```



```
0.79000      1645.5     15.357     62.358
0.80000      1666.3     16.034     65.853
left - right
SPEED  0.28280    PROCESS  4 TO  2 WEIGHT (  0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  3 WEIGHT ( -0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  6 WEIGHT (  0.0000    ,  0.50000    )
SPEED  0.28280    PROCESS  7 TO  2 WEIGHT (  0.0000    , -0.35350    )
SPEED  0.28280    PROCESS  7 TO  3 WEIGHT (  0.0000    ,  0.35355    )
SPEED  0.28280    PROCESS  7 TO  6 WEIGHT (  0.50000    ,  0.0000    )
END OF RUN on proc  -1
  energia (eV) =2082.91434
ang(grad)  ang x Ek    secc. diferenciales

0.10000E-01  20.829     15.901     4.7155
0.20000E-01  41.658     11.333     11.207
      ...
0.79000      1645.5     4.4551     62.351
0.80000      1666.3     4.6900     65.846
left- left
SPEED  0.28280    PROCESS  4 TO  2 WEIGHT (  0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  3 WEIGHT ( -0.35355    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  6 WEIGHT (  0.0000    , -0.50000    )
SPEED  0.28280    PROCESS  7 TO  2 WEIGHT (  0.0000    , -0.35350    )
SPEED  0.28280    PROCESS  7 TO  3 WEIGHT (  0.0000    ,  0.35355    )
SPEED  0.28280    PROCESS  7 TO  6 WEIGHT ( -0.50000    ,  0.0000    )
END OF RUN on proc  -1
  energia (eV) =2082.91434
ang(grad)  ang x Ek    secc. diferenciales

0.10000E-01  20.829     1.1174     4.7155
0.20000E-01  41.658     0.43665    11.207
      ...
0.79000      1645.5     157.99     62.351
0.80000      1666.3     166.45     65.846
right - 2s
SPEED  0.28280    PROCESS  4 TO  2 WEIGHT (  0.50000    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  3 WEIGHT (  0.50000    ,  0.0000    )
SPEED  0.28280    PROCESS  7 TO  2 WEIGHT (  0.0000    ,  0.50000    )
SPEED  0.28280    PROCESS  7 TO  3 WEIGHT (  0.0000    ,  0.50000    )
END OF RUN on proc  -1
  energia (eV) =2082.91434
ang(grad)  ang x Ek    secc. diferenciales

0.10000E-01  20.829     18.271     20.957
0.20000E-01  41.658     24.580     16.679
      ...
0.79000      1645.5     200.69     4.6730
0.80000      1666.3     212.17     5.0135
left- 2s
SPEED  0.28280    PROCESS  4 TO  2 WEIGHT (  0.50000    ,  0.0000    )
SPEED  0.28280    PROCESS  4 TO  3 WEIGHT (  0.50000    ,  0.0000    )
SPEED  0.28280    PROCESS  7 TO  2 WEIGHT (  0.0000    , -0.50000    )
SPEED  0.28280    PROCESS  7 TO  3 WEIGHT (  0.0000    , -0.50000    )
END OF RUN on proc  -1
```

```
 energia (eV) =2082.91434
ang(grad)   ang x Ek    secc. diferenciales

0.10000E-01  20.829     8.0345     20.957
0.20000E-01  41.658     19.581     16.679
   ...
0.79000      1645.5     209.27     4.6730
0.80000      1666.3     220.42     5.0135
total execution time =   1400.5
FORTRAN STOP
```

One aim of the program design is to provide the facility to take linear combinations of amplitudes to construct cross sections. These are read in, as in the above example, in the form of K0, KF and WEIGHT (2i3,2f10.5), where WEIGHT is a complex quantity. The amplitude to be included is that from K0 to KF in the definition of the states (adiabatic or diabatic). Combinations are summed according to the formula:

$$f = \sum_{k0}^{kf} a \cdot \text{weight}$$

It is assumed that $|\text{weight}|^2=1$ and this signals the program to begin calculation of the cross section. A further set of inputs may be given, and so on until the line K0=0, KF=0 is given. A special value of K0=999 indicates that the next line is to be treated as a title and reproduced verbatim on the output listing.

In addition to control directives in LU5 a system of files is maintained which contain molecular data. These are described in Appendix A. The only essential one is LU1 = PERUBU.DAT, which contains the energies {ε} and adiabatic couplings {M}.

Directives not used in the following sections:

The following directives are peculiar to the main program: TITLE, DOUBLE, NOPR and STOP

TITLE

the input line following this directive is reproduced verbatim on the output listing (on LU6) and may therefore serve to identify the run.

DOUBLE

Normally two files may each hold both transition amplitudes and phases. These are TANGO.DAT and COPYS.DAT. States are labelled 1 to NS in each. The action of

DOUBLE signals to certain modules (noteably total, diffxs and orbitl) that the two files are to be combined with labelling 1 to NS and NS+1 to NS+NS' respectively. This allows interference effects to be considered between states whose populations are not mixed by the collision, such as gerade and ungerade states, or positive and negative reflection symmetry in the plane. Independent calculation of their amplitudes is both faster and numerically more stable. Data can be copied between the files with the COPYS directive.

NOPR

This suppresses a certain amount of diagnostic output which would otherwise appear on stream LU4=DUMP.DAT. Further use of this directive toggles the action. The output consists of the values of x, y, z on the classical path, the unitarity achieved, and the value of the time-dependent occupation probabilities for each molecular state.

STOP

This terminates program execution when the preceeding module has finished. The files are closed and the processors are released for other users.

RADIATION

This directive signals to the following modules that data is required to be saved for use by the emission program. In particular the evolution matrix for each time step is written to the file on LU10=EVOLU.DAT.

NEWFORMAT

This directive is used to tell the program that the input file is in a special format, which might be user-defined. Currently the format must be edited into the program, which has then to be recompiled. Future versions of the code may allow a description file to be used instead which could be analysed at run time.

Implimentation of the EIKONXS charge-transfer package on the
Intel iPSC/2 at Daresbury.

To carry out large-scale calculations of the state-selective angular differential cross sections for electron capture by protons colliding with laser-excited sodium atoms the scattering package EIKONXS was ported to the Intel iPSC/2 in 1988-9. Part of this work was done with Dr.C.Courbin who visited Daresbury for one year. Some

general comments on parallelisation and introduction to the Intel hypercube are given in [16].

A natural division of the problem into independant calculations for different starting conditions (impact parameters RO), essentially a data–farming scenario, yields linear scalability with the number of processors for sufficiently large NRO. This allows the modules aprobe, and dprobe to be parallelised in a trivial, but effective, way. A similar division in angles allows diffxs to be parallelised, and division in frequencies would allow parallelisation of emission. This was however not done as it would mean substantial re–writing of the existing code. Instead the diffxs module has been parallelised by carrying out the cross section calculation for a number of different initial and final conditions simultaneously on different nodes. No significant gain is achieved with the other modules (e.g. total, diab and starks) which are therefore run sequentially on either the host or processor 0.

There is however an overhead in reading the data files as the number of processors NNODE increases. This is due to file contention. The Intel appears to read data asymptotically from the host system resource manager (SRM), that is for a large number of processors and large data file, at 0.0065 secs per kbyte per processor. The time increases for less than four processors and small files. For this reason the larger files (TANGO.DAT, COPYS.DAT, EVOLU.DAT etc.) are placed on the Intel Concurrent File System [18] which allows different blocks of the files to be read simultaneously by different processors. This results in faster access to data stored there. The input file PERUBU.DAT and remaining formatted files are kept on the host.

Whilst the Intel is a MIMD machine, access to files is akin to a virtual shared memory operation with all the inherent difficulties associated with contention and data integrity. The latter problem has had to be overcome in the present application, and we have implemented a method of semaphores on the Intel.

call semout(itype) — send semaphore from host/cube to cube/host
call semain(itype) — receive semaphore from cube/host on host/cube

Some execution times for this package are shown below.

elapsed time in seconds

7 states

number of processors	32	16	8	4	2	1	C–210	
impact parameters								
0	24.5	12.5	6.5	3.5	2.5	2.4		
64	60	83.5	149	282	553		83	125

256	167	297	575	1116	2207		332	500
512	309	581	1143	2228	4411		664	1000

14 states

number of processors	32	16	8	4	2	1	C–210	
impact parameters								
0	77	39	20.2	11.5	7.4	6.8		
64	321	542	993	1906			695	1046
256	1053	2051	3911	7589			2780	4187
512	2029	4063	7808				5560	8373

N.B. the above timings are for the Intel iPSC/2 sxvx at Daresbury running in scalar mode in early 1989 under version 3.0 of the UNIX operating system. The code was compiled on the Convex using option –O2 and therefore has some limited vectorisation for the first column, the second column shows the scalar execution times.

III Program Modules and Useage.

Module	Purpose	section
aprobe.f	adiabatic integration	1
cfsemul.f	emulate concurrent i/o on host	2
copys.f	copy workfiles LU2 to LU9	3
defln.f	calculate classical deflection functions	4
diab.f	diabatisation	5
diag.f	diagonalisation routine	6
diffxs.f	differential cross section program EIKON	7
dprobe.f	diabatic integration	8
emission.f	calculation of molecular radiation	9
gamma.f	calculation of non-unitary S-matrix	10
gaucho.f	calculation using monte-carlo method	11
graphics.f	package to do graphics using NAG	12
hamilton.f	trajectory integration	13
lzhes.f	diagonalisation of general matrix	14
main.f	main program slave nodes	15
main0.f	main program host	16
matrix.f	matrix utilities	17
orbitl.f	printing of alignment and orientation parameters	18
parameter.f	parameters controlling size of data	19
server.f	file server to access concurrent i/o	20
splinex.f	spline package (A.Salin)	21
starks.f	long-range analytic propagation	22
tfft.f	fast fourier transform for spectra	23
total.f	total cross section evaluation	24
yves.f	stores S-matrix elements and phases	25
ipsc.f	library of dummy routines for sequential version	26

1 aprobe.f

This module integrates the coupled equations (I.31) between $-z_o$ and z_o to give the complete matrix of transition amplitudes $a^i_j(-z_o,+z_o)$ in file TANGO.DAT. It requires adiabatic data which consists of energies, couplings and translation-factor matrix elements. The format of this data is described in appendix A to this document.

The method of integration is that suggested by Billing and Baer [4] which I briefly outline here. It is but one of several possible methods, a simpler one is first described (H.-J. Korsch, private communication) which is the basis of the Magnus approximation.

A formal solution of the scattering equation (11 and 13) is

$$c(t) = \exp\{-i\int_{t_o}^{t} M dt\}c(t_o) \qquad (30)$$
$$\equiv e^{-iW}c(t_o)$$

$$\equiv U(t,t_o)c(t_o)$$

The propagator U can be written as a series involving W (for which a suitable approximation must be found),

$$U = 1 - iW - \frac{W^2}{2!} + i\frac{W^3}{3!} + \frac{W^4}{4!} - \ldots \qquad (31)$$

which is the correct way to evaluate the exponential of a matrix. Truncating this series yields an approximation (W is small for small $\delta t=t-t_o$). An alternative approximation to this exponential is

$$U = (1+iW/2)^{-1}(1-iW/2) \qquad (32)$$

Implementations of this have been used in the literature with the alternating direction implicit (Peacemann-Rachford) technique [28]. A further approximation is discussed in section 5 below.

The method of Billing and Baer implemented in subroutine prop combines a second-order expression for the integral in W with a power series propagator resembling a Taylor expansion. We require values for the coupling M and its first and second derivatives in R, M' and M'', which are found using spline interpolation with subroutines in splinex.f (section 21). The matrix must be diagonalised at the start of the interval to give (all capital letters below refer to matrices, small letters to vectors and t to time):

$$D = AM(t_o)A^{-1} \text{ where A is the transformation matrix} \qquad (33)$$

using diag.f from the call to subroutine mdiag (section 6). The algorithm is then derived as follows:

$$M(t) = \sum_r \frac{\partial^r}{\partial t^r} M(t_n)\frac{(t-t_n)^r}{r!} \qquad (34)$$
$$\equiv U(t_n) + K$$

$$\text{so } i\hbar\frac{\partial c(t)}{\partial t} = (A^{-1}DA + K)c \text{ is exactly the original equation} \qquad (35)$$

By writing $b \equiv Ac$ it can be written as

$$i\hbar\frac{\partial b}{\partial t} = (D + J)b, \text{ where } J \equiv AKA^{-1} \text{ for a constant matrix A (we are free to specify this exactly).} \qquad (36)$$

Taking the diagonal parts of this equation only we can write

$$i\hbar\frac{\partial b_o}{\partial t} = (D_d + J_d)b_o \qquad (37)$$

and define $b \equiv Bb_o$ to yield a new equation for b so

$$\frac{\partial b}{\partial t} = \dot{B}b_o + B\dot{b}_o \text{ and} \qquad (38)$$

$$i\hbar\dot{B} = (D + J)B - B(J_d + D_d) \text{ where we have also defined} \qquad (39)$$

$J \equiv J_o + J_d$, and $D \equiv D_o + J_d$.

$$i\dot{B} = [D + J,B] + J_oB \quad (40)$$

Now we assume that at time t_o B is a unit matrix B=1, and that it is very slowly varying. Equation (40) is then approximated by replacing B on the right hand side by the unit matrix for all time, and for a small time interval its solution is

$$B = 1 - \int_{t_n}^{t} J_n dt \quad (41)$$

Remember that J is obtained from the Taylor expansion of the coupling, and represents the off-diagonal part of all except the first term after transforming with A. It contains an explicit time dependence of $(t-t_n)^r$ for each of its terms which can be integrated explictly to any order we wish to take. We now look at the values of B at the start of an interval, with t_n being the centre point of the interval so $(t-t_n)^r = (\pm 1)^r(\Delta t/2)^r$ where Δt is the time step chosen for the computation. This gives us $B(t_o)$ and $B(t)$. In a similar way it is possible to solve for $b_o(t) = E(t)b_o(t_o)$. Combining all the above definitions the propagator is

$$U(t,t_o) = A^{-1}B(t)E(t)B^{-1}(t_o)A. \quad (42)$$

Explicitly I have programmed this to include up to the second derivative in M, found by spline interpolation over the numerical coupling data and explicit values for the multiplicative constants which include the nuclear coordinates, velocities and accelerations computed in hamilton.f (section 13). This is done in subroutine dercup. The algorithm in subroutine prop is as follows, writing F + G = J:

$$AM(t+\Delta t/2)A^{-1}=D$$

$$F=A\frac{dM}{dt}A^{-1}$$

$$G = \frac{1}{2}A\frac{d^2M}{dt^2}A^{-1}$$

$$F_o=F(1-\delta),\ G_o=G(1-\delta)$$

$$B(t_o)=1-\frac{i}{\hbar}[\frac{1}{8}F_o(\Delta t)^2-\frac{1}{24}G_o(\Delta t)^3]$$

$$B(t)=1-\frac{i}{\hbar}[\frac{1}{8}F_o(\Delta t)^2+\frac{1}{24}G_o(\Delta t)^3]$$

$$E_1^k(t)=\exp\{-\frac{i}{\hbar}D_1^k\Delta t+\frac{1}{12}G_1^k(\Delta t)^3]\}\delta_1^k \quad (43)$$

The original derivation of this algorithm is given in reference [4] in the case of real symmetric matrices.

Comments on the algorithm and choice of time step.

In practice the equations are stepped in pseudo-spatial coordinate z = vt. This is a distance along the trajectory if the velocity is constant. Sometimes, for small values of impact parameter ρ, *i.e.* strong collisions, the unitarity is poor; it should normally be

around 1 ± 10^{-6}. The deviation is due to the time step being too large for these strong collisions where everything may occur near z=0. Note that at present the program uses a fixed step rather than a variable one, this has proven most efficient to implement but is not ideal for all situations.

Aprobe is the most important module of the program package. Example control data on stream LU5 was shown above.

The first line of input is a cutoff parameter ALPHA (f12.5) defined by Errea et al. [20] which modifies the form of the ETF (19) to

$f = \frac{Rz}{(R^2+\alpha^2)}$, which may be used for optimisation purposes [21].

The next two lines are NV and V (i3/f10.5). At present the program can only cope with one relative velocity (in atomic units), so NV=001. The next lines control the spacing of the impact parameters for which the integration is carried out. Firstly NGROUP and RO(1) (i3,f10.5) then a number of lines NGROUP each of (i3,f10.5) to add NRG impact parameters of spacing ROSTEP. The pseudocode to read these lines is

```
...
read(5,'(i3,f10.5)')ngroup,ro(1)
j=1
do i=1,ngroup
read(5,'(i3,f10.5)')nrg,rostep
do k=1,nrg
j=j+1
ro(j)=ro(j-1)+rostep
end do
end do
...
```

The following lines are read on f10.5 format and are RSTART, RSTOP, ZSTEP, reduced mass and RPHAS. Integration is carried out from ZSTART to ZSTOP where

zstart = sqrt(rstart*rstart-rho*rho)

zstop = sqrt(rstop*rstop-rho*rho)

if these conditions can be satisfied. If not the starting conditions are changed and a warning message is printed. The use of RPHAS is described in [1] with which I assume the reader is familiar.

The next line of input is NCOL1, NCOL2 (2i3) which controls the trajectory; its action is described in section 13. The next lines are EFIN (6f10.5) and ME (40i3) which are also as in reference [1]. The last two lines of input are a logical variable (l1) 'T' or 'F' which controls printing of output and a constant (f10.5) which is currently unused but available for future purposes. Output may be printed on LU4 = DUMP.DAT

giving values of the probability matrix, unitarity and nuclear coordinates at each time step. This may prove useful as an initial check on the integration.

The following subroutines are contained in file aprobe.f

aprobe.f:1: SUBROUTINE TANGO(IPROG)
aprobe.f:140: SUBROUTINE CALINE(JJ)
aprobe.f:185: SUBROUTINE TRAB(ME,N)
aprobe.f:209: SUBROUTINE DIEGO(NS,ZMAX,ZPHAS,PHI,EFIN,PHI2)
aprobe.f:233: SUBROUTINE DERCUP(NDIM,Z,U,DUDR,D2UDR2)
aprobe.f:311: SUBROUTINE MDIAG(NDIM,N,D,A,T,AT)
aprobe.f:345: SUBROUTINE PAMPA (NV, VE, NRO, RO, NS, RIN, RM, HIN, EPS, RPHAS, EFIN, ME)
aprobe.f:487: SUBROUTINE PROP(NS,Z1,C,DEL,EPS,S,SP,DUMMY,FIN)
aprobe.f:571: SUBROUTINE CZERO(NDIM2,U)
aprobe.f:577: SUBROUTINE VMULTD (NDIM, U, AV1, C1, AV2, C2, AV3, C3, AV4, C4, AV5, C5, NS, NC)
subroutine sort2(...)
subroutine permvec(...)

2 cfsemul.f

This program is a set of routines which pass messages to the file-server program server.f. They are called from the host to read files stored on the Intel Concurrent File System [18], particularly files TANGO.DAT and COPYS.DAT which can not be read directly on the host. A full description appears in [16].

The following subroutines are contained in file cfsemul.f

cfsemul.f:13: subroutine open(lu)
cfsemul.f:20: subroutine rewind(lu)
cfsemul.f:27: subroutine read(lu,buffer,nbytes)
cfsemul.f:37: subroutine write(lu,buffer,nbytes)
cfsemul.f:48: subroutine close(lu)
cfsemul.f:54: subroutine copy4(idim,imax,jmax,ain,buffer)
cfsemul.f:66: subroutine copy8(idim,imax,jmax,ain,buffer)
cfsemul.f:78: subroutine un4(idim,imax,jmax,ain,buffer)
cfsemul.f:90: subroutine un8(idim,imax,jmax,ain,buffer)

3 copys.f

This program copies the contents of file TANGO.DAT into the workfile COPYS.DAT. It is required by the module starks for instance which transforms the contents of COPYS.DAT and writes its results in TANGO.DAT. It may also be used to run the package for different symmetries and combine them using the DOUBLE directive with modules total, orbitl and diffxs.

Two directives are COPY and UNCOPY which copy data from LU=2 to LU=9 or the reverse respectively.

The following subroutines are contained in this file

copys.f:1: SUBROUTINE COPYS(NIN,NOUT)

4 defln.f

This program runs with the aprobe.f module and yields classical deflection functions calculated on the different potential energies. The deflection function is evaluated from the formula:

$$\Theta(E,b) = \pi - 2b\int_a^\infty \frac{dR}{R^2\sqrt{[1-b^2/R^2-V(R)/E]}} \qquad (44)$$

where a is the point of closest approach, and V(R) is the average potential energy of states selected using the NCOL parameters.
The integral is carried out using Gauss-Mehler quadrature. This is discussed in [19], [9] and references therein.

This program is invoked with the directive CLASSICAL which prevents the full propagation from being carried out. The directive should be put before the directive to APROBE or DPROBE.

The following subroutines are contained in this file

defln.f:1: SUBROUTINE defln(efin,kr)
defln.f:59: SUBROUTINE fct(XR,XF,XDERF,IERR)
defln.f:80: SUBROUTINE RTNI(X,F,DERF,FCT,XST,EPS,IEND,IER)
defln.f:113: subroutine deflec(xteta,dtdb,rturn)
defln.f:167: SUBROUTINE SETA(ETA,RT,YL,NGM)
defln.f:201: subroutine gturn(rturn)

5 diab.f

This program reads adiabatic potential and coupling data on LU1 and transforms it to produce diabatic data which is written on LU8. It is used before running the module dprobe or the program QUANTXS [23]. The transformation is performed with the method outlined in [5, 11 and 17].

A novel modification is made to the usual method to allow for the fact that the radial coupling may not be zero as $R \to \infty$. We include the ETF correction in equation (45) below, which ensures correct long-range behaviour and convergence of the integration.

The diabatic transformation is defined from the coupled equations (28) by taking just the radial coupling terms (those in dR/dt) and leaving the rotational coupling terms (those in dγ/dt) for the main integration. We thus need to solve

$$\frac{d\,l}{dR}^{m}_{k} = [\langle\chi_m|-i\tfrac{\varepsilon}{v}n\frac{\partial}{\partial R}|\chi_n\rangle - \langle\chi_m|z\frac{\partial}{\partial z}|\chi_n\rangle/R]l^{n}_{k};\ \forall k \qquad (45)$$

This assumes that coupling terms vanish as $R\to\infty$ as required

$$\frac{dl}{dR} = -Al \text{ in matrix form} \qquad (46)$$

(A is not the same matrix as in the earler section!).

The formal solution to this equation is

$$l(R+h) = e^{\mathscr{A}h}l(R) \qquad (47)$$

where $\mathscr{A}$ must be determined approximately, and h is a small step in distance. Expanding

$$l(R+h) = (1 + \mathscr{A}h + \mathscr{A}^2\frac{h^2}{2!} + \mathscr{A}^3\frac{h^3}{3!} + \ldots)l(R) \qquad (48)$$

The Taylor series is

$$l(R+h) = l\big|_R + h\frac{dl}{dR}\Big|_R + \frac{h^2}{2!}\frac{d^2l}{dR^2}\Big|_R + \ldots \qquad (49)$$

by comparison after substituting (46) and its derivatives into (49) to eliminate l

$$\begin{aligned}\mathscr{A} &= -A\\ \mathscr{A}^2 &= A^2 - A'\\ \mathscr{A}^3 &= AA' + 2A'A - A'' - A^3\end{aligned} \qquad (50)$$

A better approximation is found from the formula correct to the 5th order given by Gargaud [19].

$$\begin{aligned}\mathscr{A} = -A &-A'\frac{h}{2!} - \{A'' + \tfrac{1}{2}[A,A']\}\frac{h^2}{3!} + \{A''' + [A,A'']\}\frac{h^3}{4!}\\ &-\{A''' + \tfrac{3}{2}[A,A'''] + [A,A'']\}\frac{1}{6}[[A,A''],A] - \frac{1}{2}[A',[A,A']]\end{aligned}$$

$$\frac{1}{6}[[[A,A'],A],A]\}\frac{h^4}{5!} \qquad (51)$$

In practice only terms up to h^3 are included which requires second and third derivatives of the coupling, the latter being found by a central difference formula.

Equation (51) is used to step backwards from $R=R_{max}$ to R=0 using very small negative values of h.

$$l(R-h) = (1-\mathscr{A}h + \mathscr{A}^2\frac{h^2}{2!} - \mathscr{A}^3\frac{h^3}{3!})l(R) \qquad (52)$$

The resulting matrix l(R) is used to transform the coupling according to

$$\varphi = \chi l; \qquad c = ld \qquad (53)$$

equation (11) yields

$$i\frac{\partial c}{\partial t} = Mc \text{ therefore } \qquad i\frac{\partial d}{\partial t} = (l^{-1}Ml - il^{-1}\dot{l})d \qquad (54)$$

the coupling in the diabatic equations is defined to be

$$M^d = l^{-1}Ml - il^{-1}\frac{\partial l}{\partial t}. \qquad (55)$$

We note that applying this to the radial coupling terms gives zero by definition. All remaining diabatic coupling arises from rotational coupling and energies.

An example of input to this module is given below, it resembles input to module aprobe. Lines are ALPHA (f10.5), EFIN (6f10.5), ME (40i3), H(f10.5), PRFLG (l1) and RMAX (f10.5).

Finally, note that the diabatic quantities are quite unphysical and appear to exhibit very peculiar behaviour in some cases of strong coupling.

The following subroutines are contained in the file

diab.f:1: SUBROUTINE DIAB
diab.f:104: SUBROUTINE PAMPAD (NV, VE, NRO, RO, NS, RIN, RM, HIN, EPS, RPHAS, EFIN, ME)
diab.f:211: SUBROUTINE PROPR(NS,Z1,C,DEL,EPS,S,SP,DUMMY,FIN)
diab.f:281: SUBROUTINE DERCUD(NDIM,R,U,DUDR,D2UDR2)
diab.f:320: SUBROUTINE PROPD(NS,Z1,C,DEL,EPS,S,SP,DUMMY,FIN)
diab.f:352: SUBROUTINE ZERO(NDIM2,U)

Example of input file on LU5 to control module diab

```
2 diab
DIAB
0.                    alpha
-0.18848  -0.125   -0.125   -0.11051  -0.07159  -0.125
-0.11155
```

```
 0 0 0 0 0 1 1
0.1
F
50.0
STOP
```

6 diag.f

This is the routine which performs diagonalisation of real symmetric matrices. It is called from the programs aprobe.f and dprobe.f via the routine mdiag which transforms a hermitian matrix into a real symmetric supermatrix as follows:

Consider an eigenvalue equation involving a hermitean matrix, remembering that it has real eigenvalues

$$MX = \lambda X, \qquad M=u+iv, \qquad X=x+iy \tag{56}$$

$$(u+iv)(x+iy) = \lambda(x+iy) \tag{57}$$

$$ux - vy = \lambda x$$

$$uy + vx = \lambda y \tag{58a}$$

$$\begin{bmatrix} u & -v \\ v & u \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix} \tag{58b}$$

The matrix $\begin{bmatrix} u & -v \\ v & u \end{bmatrix}$ is the supermatrix which is passed to DIAG.

A set of 2*NS doubly-degenerate eigenvalues and vectors are obtained.

The following subroutines are contained in this file

diag.f:1: SUBROUTINE DIAG (M,N,A,D,X)

7 diffxs.f

This program calculates differential cross sections using the matrix of transition amplitudes and phases contained in the file TANGO.DAT. It uses the CPC program EIKON described in [2], which is only correct for a straight-line trajectory. For a more general analysis see reference [31]..

The algorithm used by this program is based on the partial-wave formula

$$\frac{d\sigma}{d\Omega}(\Theta) = |4\pi\sum_l(2l+1)(S_j^{i(l)} - \delta_j^{i(l)})P^l(\cos\Theta)|^2 \tag{59}$$

in the semiclassical limit

The partial wave is related to the impact parameters $l=\mu v\rho$ (μ is the reduced mass) and $S_j^i(l)$ is obtained by 5-point Gaussian interpolation of the amplitudes. This is

numerically stable since the amplitude c_j^i and its related phases v_i and v_j calculated in routine diego (see [1]), are stored and interpolated separately and the Coulombic phase from the electronic core repulsion is added explicitly as shown below.

The quantities calculated by the program are:

$$\frac{d\sigma}{d\Omega}(\Theta) = |f(\Theta)|^2 \tag{58}$$

$$f(\Theta) = ik\int_0^\infty J_M(2k\sin(\Theta/2)\rho)c_j^i\exp(-\frac{i}{v}[v_j(|z_p|)+v_i(|z_p|)+\alpha_p])\,; \quad k=\mu v \tag{61}$$

where the approximation for α is

$$\alpha(\rho) = (\zeta_j+\zeta_i)\ln(\tfrac{1}{2}[1 + \sqrt{(1 + \rho^2/z_p^2)}]) - 2Z_AZ_B\ln(\rho) \tag{62}$$

in which ζ_j and ζ_i are the asymptotic effective charges of the system in its jth or ith state; both zero for ion-neutral collisions. Z_A and Z_B are the effective charges of the two ionic cores seen by the active outer electron, e.g. 1.0 in the case of a sodium atom. z_p is the distance at which the scattering phase quantities v_i, v_j were evaluated as given by ZPHAS in the input to aprobe and dprobe.

Input to the diffxs module is as follows: MANG (f10.5) number of angles to do, THETA(1) (f10.5) first angle, THSTEP (f10.5) increment, M, JBP (2i3), ZA, ZB, ZETAI, ZETAJ (f10.5) quantities defined in [2], RMASS (f10.5). JBP is an index which, if set to 1 allows an approximate expression for the Bessel function to be used, otherwise set to 2 (see [2]).

Further input lines refer to linear combinations of the amplitudes stored in TANGO.DAT and COPY.DAT to be used in evaluating the dcs. They are read as follows K0, KF, MDE, WEIGHT (3i3,2f10.5) where WEIGHT is a complex quantity. This was described in section II. MDE is defined in [2] and is zero for states of the same symmetry and 1 for σ-π transitions. An END directive is optional.

The following subroutines are contained in this file

diffxs.f:1: SUBROUTINE DIFFXS

diffxs.f:122: SUBROUTINE EIKON (ZA, ZB, ZETI, ZETJ, RPHAS, V, VMU, MANG, ANG, MDE, JBP, BMP)

diffxs.f:290: SUBROUTINE BESSEL(MDE,X,BES)

Example of input stream LU5 to control diffxs [22, 30]

16diffxs

```
 Na-H+ 14 estados, 23/6/89
DOUBLE
EIKON
160
0.01
0.01
  0  2
1.
1.
0.
0.
1.0426
999
 3s to H(2p)
  1  2  0  0.707
  1  3  0  -0.707
999
 3s to 2s
  1  2  0  0.707
  1  3  0  0.707
999
 3p to H(2p)
  4  2  0  0.707
  4  3  0  -0.707
999
 3p to 2s
  4  2  0  0.707
  4  3  0  0.707
999
 3p pi to H(2p)
 11  2  1  0.707
 11  3  1  -0.707
999
 3p pi to 2s
 11  2  1  0.707
 11  3  1  0.707
999
 3s to 2p pi
  1 10  1  1.0
999
 3p sigma to 2p pi
  4 10  1  1.0
999
 3p pi to 2p pi
 11 10  0  1.0
999
 3p pi- to 2p pi-
 25 24  0  1.0
999
px px + py py
 11 10  0  0.707
 25 24  0  0.707
999
px pz - pz px
```



```
 11  2  1  0.5
 11  3  1  -0.5
  4 10  1  -0.707
999
px pz + pz px
 11  2  1  0.5
 11  3  1  -0.5
  4 10  1  0.707
  0  0
STOP
```

Output from the diffxs module is shown in the listing at the end of chapter II. A graphical example is shown in the figure following section 18, this is the differential cross section for $Na(3p_{+1}) + H^+ \rightarrow Na^+ + H(2p_{+1})$ circularly polarised states and its convolution with an experimental window of detection of finite resolution.

8 dprobe.f

This program integrates the set of coupled equations using the diabatic data stored in file LU8 = DIAB.DAT which might have been produced from module diab.f as described above. It uses the same method as aprobe, and takes the same input. In general the step in z may be some 1.5 to 2 times larger or even more with many tight crossings [27].

A further peculiarity of the coupled equations for the diabatic case is their symmetry to time reversal. This symmetry is absent in the adiabatic equations which must therefore be integrated numerically from $-z_o$ to $+z_o$. The diabatic equations are integrated from $-z_o$ to z=0 and then the inverse of the propagator is used to take the solution from z=0 to z_o. This can only be done for straight-line trajectories (NCOL1=NCOL2=0) otherwise the full integration is needed.

From (54)

$$i\frac{\partial d}{\partial t} = I^{-1}Bd \tag{63}$$

where I(R) is time symmetric and

$$B = \langle\chi_m|\epsilon_n + i\frac{d\gamma_i}{dt\hbar}L_y|\chi_n\rangle - i\frac{d\gamma}{dt}\langle\chi_m|x\frac{\partial}{\partial z} + z\frac{\partial}{\partial x}|\chi_n\rangle + 1.5\left(\frac{d\gamma}{dt}\right)^2\langle\chi_m|x^2|\chi_n\rangle \tag{64}$$

which is also time symmetric. Noting the reversal of the molecular axis around z=0 it is possible to build the propagator matrix

$$d(0) = U^d(0,z_o)d(-z_o) = \Pi_{z=-z_o}^{0} U^d(z+h,z)d(-z_o) \tag{65}$$

$$d(z_o) = U^d(z_o,0)d(0) = \Pi_0^{z_o} U^d(z+h,z)d(0) \qquad (66)$$

$$\text{and } U^d(z+h,z) = U^d(-z,-z-h). \qquad (67)$$

An overall speedup of around 3 times is possible using the dprobe module as compared to aprobe, more if there are tight crossings as in the HeH^+ system [27].

The following subroutines are contained in file dprobe.f

dprobe.f:1: SUBROUTINE DPROBE

dprobe.f:79: SUBROUTINE DERCU2(NDIM,Z,U,DUDR,D2UDR2)

Example of input on LU5 to control dprobe. The last parts of this example refer to the total and orbitl modules described in sections 24 and 18 below

```
32dprobe
NOPR
TITLE
****** NA-H+ 7 ESTADOS, rmax=30.  nro=6 ** -2*phi2, efin en h
DPROB
0.                    alpha
001
0.2828
004 1.
002 0.3
001 0.707
001 0.7
001 1.
-30.
30.
1.
1.0426
30.
 0 0
-0.18848  -0.125    -0.125    -0.11051  -0.07159  -0.125
-0.11155
 0 0 0 0 0 1 1
F
0.0
TOTAL
999
 **** Transicion NA(3S) ---> H(2P) *****
 1 2  0.707
 1 3  -0.707
 0 0
ORIENT
PROBABILITY
999
right circularly polarised H(2p) from right circularly polarised Na(3p)
```



```
 4 2 0.35355
 4 3 -0.35355
 4 6          0.5
 7 2          0.35355
 7 3          -0.35355
 7 6 -0.5
999
right - left
 4 2 0.35355
 4 3 -0.35355
 4 6          -0.5
 7 2          0.35355
 7 3          -0.35355
 7 6 0.5
999
left - right
 4 2 0.35355
 4 3 -0.35355
 4 6          0.5
 7 2          -0.35355
 7 3          0.35355
 7 6 0.5
999
left- left
 4 2 0.35355
 4 3 -0.35355
 4 6          -0.5
 7 2          -0.35355
 7 3          0.35355
 7 6 -0.5
 0 0
END
STOP
```

9 emission.f

This program uses the diabatic evolution matrix U^d defined above which is stored in the file attached to stream LU10, as well as the transition dipole moments on LU11 and LU12 to evaluate the optical dipole emission spectrum for radiative transitions during the collision. This is a uniform method for both the line-centre and far-wing spectra and accounts for dynamical transitions in the gas phase. The method used is a first-order theory outlined by Briggs and Dettman [14] which requires a numerical fast-Fourier transform as described by Pacher [15] and Terao [12]. We have used the program of Temperton [13] to do this. Note that diab and dprobe must be called before emission in the current implementation.

The expression for the emission spectrum is

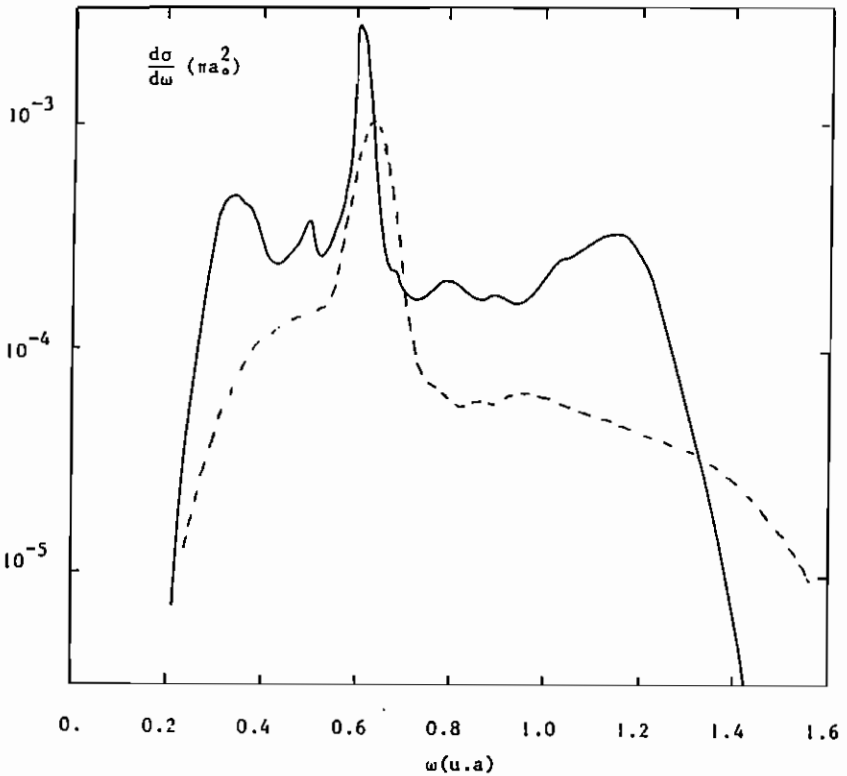


Fig. 3

$$f_\alpha = i\sqrt{\left(\frac{2\pi}{\omega v}\right)}\int_{-\infty}^{\infty}\sum_{n,m} c_n^{*+} f_{(t)} c_i^{-m}{}_{(t)} \langle \chi_n(t)|p.\varepsilon^{(\alpha)}|\chi_m(t)\rangle e^{i\omega t} dt \qquad (68)$$

$$\frac{d\sigma}{d\omega} = \frac{4\omega}{3c^3}\frac{\omega v}{2\pi}\sum_{\alpha=x,y,z}\int_0^{\infty}\rho d\rho\,|f_\alpha|^2 \qquad (69)$$

summing over the three components of the p.ε operator, where c^+ is the incoming amplitude, U(t,−∞), and c^- is the outgoing amplitude (time steps multiplied in the reverse order, see section 8), U(+∞,t).

The following subroutines are contained in this file

emission.f:339: SUBROUTINE SUM(IZN,RM,RHO,V,Z,NF)
emission.f:402: SUBROUTINE DIADIP(IRR)
emission.f:468: SUBROUTINE TRABD(ME)
emission.f:497: SUBROUTINE CALIND(RO,RO2,KMIN,KMAX)
emission.f:545: SUBROUTINE CONDIF
emission.f:538: COMPLEX FUNCTION ZCEXP(PHI)

At the time of writing this mocule is not fully interfaced to the rest of the package, but may be run as a separate program. Some typical output is plotted in figure 3, which is the spectrum produced during a collision of B^{5+} with H(1s) yielding H^+ and B^{4+} by electron capture, and finally decaying to B^{4+}(n=3) possibly by photon emission. We emphasise that the excited states are short lived, and photon emission occurs during the collision from molecular states, except for the central peak at ω=0.63 a.u. which is characteristic of an atomic line, and not properly resolved here. Large variations of the profile around the wings, and extension into the classically forbidden region is seen as the velocity is varied from v=0.1 a.u. to v=0.3 a.u.

10 gamma.f

This is not finished ..... (Terao and Allan, 1990)

The following subroutines are implemented

gamma.f:1: SUBROUTINE gamma(IPROG)
gamma.f:147: SUBROUTINE CALINE(JJ)
gamma.f:191: SUBROUTINE TRAB(ME,N)
gamma.f:214: SUBROUTINE DIEGO(NS,ZMAX,ZPHAS,PHI,EFIN,PHI2)
gamma.f:237: SUBROUTINE DERCUP(NDIM,Z,U,DUDR,D2UDR2)

gamma.f:314: SUBROUTINE MDIAG(NDIM,N,D,A,T,AT)
gamma.f:347: SUBROUTINE PAMPA (NV, VE, NRO, RO, NS, RIN, RM,
HIN, EPS, RPHAS, EFIN, ME)
gamma.f:488: SUBROUTINE PROP(NS,Z1,C,DEL,EPS,S,SP,DUMMY,FIN)
gamma.f:571: SUBROUTINE CZERO(NDIM2,U)
gamma.f:577: SUBROUTINE VMULTD (NDIM, U, AV1, C1, AV2, C2, AV3,
C3, AV4, C4, AV5, C5, NS, NC)

11 gaucho.f

This is not finished ..... (ideas only Allan, 1985)

It attempts to solve the coupled equations by a generalised Monte–Carlo surface–hopping technique, which won't work due to a theoretical impasse in defining the phase change when a transition occurs (re. Landau–Zener phases...)

The following subroutines are implemented
gaucho.f:1: SUBROUTINE GAUCHO (NV, VE, NRO, RO, NS, ZM, HIN,
EPS, ZPHAS, EFIN, ME, IPAR)
gaucho.f:138: SUBROUTINE FCURVE(Z,C16,DC16,ER)
gaucho.f:218: SUBROUTINE HCOUPG (DPXDT, DPYDT, DPZDT, C, NT,
ER, VOPTV, RACT, ISTATE)
gaucho.f:252: SUBROUTINE DIFSY2(N,Z,C,HO,EPS,S,SP,FCOUP,FIN)
gaucho.f:351: SUBROUTINE RSTART(I,J,K,L)
gaucho.f: FUNCTION UNI()

12 graphics.f

This produces the data contained in the various files in a form suitable for graphical output, and calls the graphical routines if such are available. I have used the VOGLE and VOPLE graphics libraries which are freely available [32] and mimic the Silicon Graphics software. An alternative way to prepare data for graphics is to use the perubu.f program described in chapter IV.

The first line of input after the GRAPHICS directive is the number of the input file unit (i2). At present this may be 1 (adiabatic energies), 8 (diabatic energies), 2 (probabilities) or 3 (differential cross sections). The column of data to be plotted on the ordinate is then given (i1) followed by the number (i1) designating the size of paper to

be used and 1 if all is OK so far. Next the minimum and maximum x and y values are given (f10.5) for plotting the axes, the number of tick marks for x and y axes (2i2) and a scaling factor and vertical shift in the y direction (f10.5). If lettering on the axes is required a letter T is given (l1) otherwise F. If required this is followed by x and z axis labels and a number for each being the number of characters to plot (i2). 3 digits follow identifying the colour of pen and line style to use, they currently have no meaning and had best be left as in the example (1 1 1). A zero if spline interpolation is required between the point, otherwiseee the program will prompt for a symbol type, and finally a zero to draw another curve on the same axes, 1 for a new y axis, or 2 to stop.

Example input to graphics module.

```
1 graphics
TITLE
   test of graphics routine on Convex
NOPR
GRAPHICS
 8
2
5
1
0.0
25.0
-1.0
0.0
 6 5
1.0
0.0
T
R [ao]
 6
energy [hartree]
16
 1 1 1
0
0
```

```
3
1.0
0.0
 1 1 1
0
2
0
STOP
```

The following subroutines are implemented on the NAG library calling sequence

```
graphics.f:1:     subroutine load
graphics.f:15:     subroutine gdprob
graphics.f:40:     subroutine gaprob
graphics.f:92:     subroutine gdiff
graphics.f:116:     SUBROUTINE gprob
graphics.f:210:     subroutine plot
graphics.f:426:     SUBROUTINE START
graphics.f:454:     SUBROUTINE RAXIS(XP,YP,BCD,NCHAR,AXLEN,ANGLE
graphics.f:553:     SUBROUTINE SORT2(K,RR,ARA)
graphics.f:589:     SUBROUTINE TAXIS(XP,YP,BCD,NCHAR,AXLEN,ANGLE
graphics.f:653:     SUBROUTINE SETUU
graphics.f:661:     SUBROUTINE SETIN
graphics.f:672:     SUBROUTINE FRAME
```

13 hamilton.f

This is the program which computes the classical trajectory, or straight line if there isn't one. It is controlled by the values of variables NCOL1 and NCOL2 read in by the calling module. It solves Hamilton's equations for the nuclear coordinates x,y,z, and momenta px, py, pz variables taking several prescriptions for the internal energy of the collision system as a function of R. These prescriptions are as follows:

NCOL1=NCOL2 — use adiabatic energy E(NCOL1), or obtain it from the diabatic quantities if routine crder has been called from dprobe.

NCOL1=NCOL2=0 — use a straight-line $R^2 = b^2 + v^2t^2$

NCOL1=n NCOL2=0 — use an average of all internal energies weighted by the probabiity of occupation of the state at R(t). Starting condition is in state n with

corrected asymptotic kinetic energy. Therefore the nth column of the evolution matrix is used to compute the probabilities. This is slightly more complicated in the case of diabatic quantities and corrections are made to include the couplings too.

NCOL1=n NCOL2=m — use a trajectory based on just the average potential of states n and m.

These average potential descriptions have been discussed in the literature [7–9]. They are not suitable for detailed calculation, but may help decide if a straight–line description is unsuitable. In that case I recommend using the diabatic transformation and passing the data into the QUANTXS program described in a separate document [23]. This is normally only necessary for collision energies in the order of 10eV, but is system dependent. For the case of Coulomb charges large effects may be found, see [17].

The following subroutines are contained in file hamilton.f

```
hamilton.f:1:     SUBROUTINE DIFSY1(N,X,Y,H0,EPS,S,F,FIN)
hamilton.f:112:     SUBROUTINE CRDER(ZN,RR,DRDZ,D2RDZ2,D3RDZ3,ROSR2)
hamilton.f:184:     SUBROUTINE HCOUP(T,Q,DQ)
hamilton.f:244:     SUBROUTINE LINFIT(NR,RR,TR,U,DUDR,D2UDR2,R)
```

14 lzhes.f

This routine attempts to diagonalise any general complex square matrix. If it can't a warning can be given. It is used in the program gamma for the case of a non–hermitian coupling matrix.

The following subroutines are contained lzhes.f

```
lzhes.f:1:     SUBROUTINE LZHES(N,A,NA,B,NB,X,NX,WANTX)
lzhes.f:138:     SUBROUTINE LZIT (N, A, NA, B, NB, X, NX, WANTX, ITER,
EIGA, EIGB)
```

15 main.f

This is the main program which controls execution of the nodes.

The following subroutines are contained in the file in addition to the main node program

main.f:155: subroutine semout(itype)
main.f:162: subroutine semain(itype)

16 main0.f

This is the main program which runs on the host computer.

The following subroutines are contained in the file in addition to the main host program

main0.f:202: SUBROUTINE YVES(NS,iproc)
main0.f:250: SUBROUTINE TOTAL
main0.f:400: SUBROUTINE SIMINT(N,X,Y,VAL)
main0.f:425: SUBROUTINE COPYS(NIN,NOUT)
main0.f:458: subroutine semout(itype)
main0.f:464: subroutine semain(itype)
main0.f:470: SUBROUTINE DIFFXS

17 matrix.f

This file contains a set of matrix utilities, such as matrix multiplication and inversion.

The routines are

matrix.f:2: SUBROUTINE CMATP(NS,C,A,B,NDIM)
matrix.f:16: SUBROUTINE CMATIN(NORDER,A,DET,NDIM)
matrix.f:86: SUBROUTINE MATMUL(NS,C,A,B,NDIM)
matrix.f:100: SUBROUTINE MATINV(NORDER,A,DET,NDIM)
matrix.f:170: SUBROUTINE CMAT13(NS,C,A,B,NDIM)
matrix.f:184: SUBROUTINE CMAT23(NS,C,A,B,NDIM)

18 orbitl.f

The program orbitl is one of the more complex programs to describe because it has arisen from the need to compare results with experimental data, and particularly to look at orbital alignment and orientation, and special combinations of differential cross

sections and their convolution over finite windows of observation [22, 29, 30].

The program expects one of five further directives which are now described in turn. These are PROBABILITY, DCS, CONVOLUTION, ADD and END.

PROBABILITY

Output is the probability for a transition with specific initial and final states as a function of impact parameter [e.g. 22].

An example of using the PROBABILITY directive is contained in the example of the dprobe module in section 8 above. Input should be finished with an END directive. The lines of input are otherwise identical to those given to the diffxs module except that the index MDE must *not* be given.

A number of special files are created, or appended if they already exist by this directive. The file names are PROBn.DAT where n is an integer numberindicating that the results are for the nth initial and final conditions reference in the control file. It is possible in this way to run the program several times with different velocities and build up a set of files which contain $P_f^i(b,v)$. This is useful for doing surface plots of the probability function as shown for instance in figure 4 which is for $Na(3p_{+1}) + H^+ \rightarrow Na^+ + H(2p_{+1})$.

DCS

Output is a differential cross section for transitions between given combinations of initial and final states of a non-isotropic target [22]. It is built from the state-to-state data in file LU3=DIFFXS.DAT (see example input in section 7).

The data given to this module is as follows. Firstly a single variable PHI (f10.5) tells it the azimuthal angle of observation in degrees (often zero); the polar angles are taken from the cross section data file on LU3, it is assumed that diffxs has been run previously. The following lines each give an index number and a complex weight (i2, 2f10.5) for the corresponding amplitude. The differential amplitudes are read from unit LU3 which has been produced by the diffxs program (see above, section 7). A number of cross sections for different initial and final conditions have been stored in this file. An index line must be read for each of these telling ORBITL how to use it. These indices are as follows:

0 — terminate data input and go on to calculation
1 — add to coefficient to be multiplied by 1.0 in formula
2 — add to coefficient to be multiplied by cos(phi)

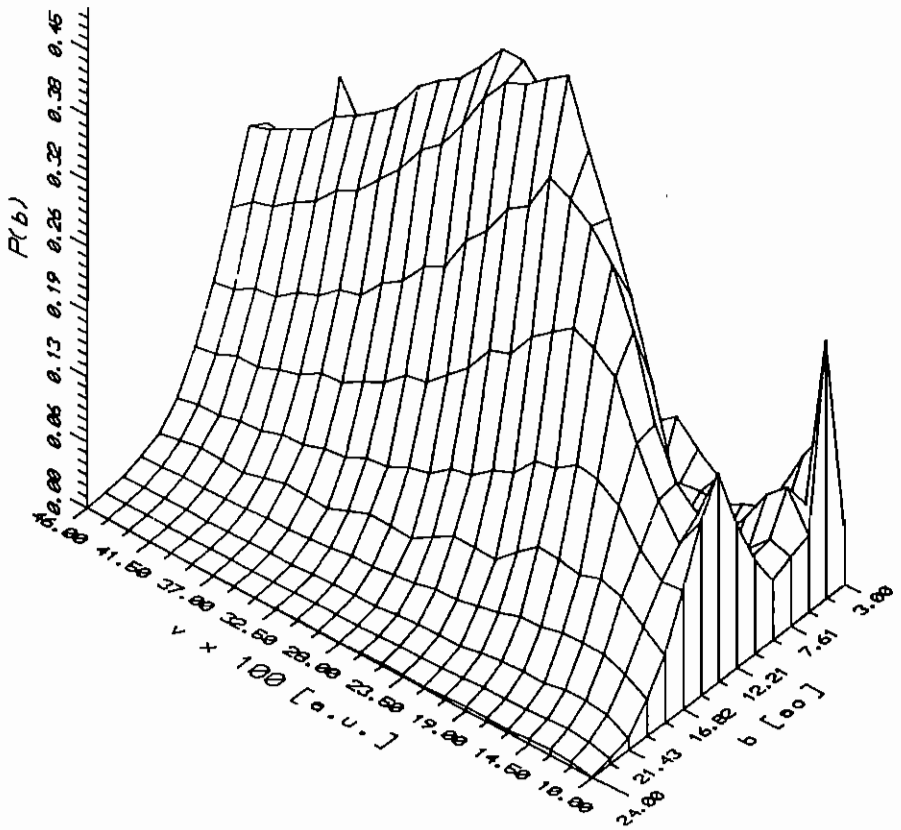


Fig.4

3 — add to coefficient to be multiplied by sin(phi)
4 — add to coefficient to be multiplied by cos(phi)sin(phi)
5 — add to coefficient to be multiplied by $\cos^2$(phi)
6 — add to coefficient to be multiplied by $\sin^2$(phi)
>6 — discard this cross section and go on to read next

The formula for the cross section is thus dcs= $|f(1)+ f(2)\cos(phi)+ f(3)\sin(phi)+ f(4)\cos(phi)\sin(phi)+ f(5)\cos2(phi)+ f(6)\sin2(phi)|^2$ which is sufficient for describing scattering by up to p orbitals aligned relative to the collision plane [30]. This is computed for each polar angle. Following this input a directive ADD allows the procedure to be repeated and the result added to the first cross section at each angle. A directive END returns control to the calling program.

CONVOLUTION

Output is the differential cross section convoluted with a Gaussian function which represents an experimental resolution in a given azimuthal angle of observation and range of polar angles determined by the input data. Derivation of this formul is given in [30].

The data following this directive is the same as that for DCS with the addition of two lines. The first contains a single variable SIGMA (f10.5) which is the half–width of a gaussian in degrees representing the experimental window of observation with cylindrical symmetry about the direction of observation (that is all theta angles in the data and the given azimuthal angle phi). The second contains the variables NAN, NG and THCUT (2i5, f10.5) which tell the program the number of Gaussian points to use for the integration in angles θ' and Φ' in polar coordinates relative to the angle of observation taken as z direction, and a cutoff angle outside which the integration in θ' has no further contribution. These lines should precede those given to the DCS module. An example is shown below, in this example the cross sections are those yielded by the example input to diffxs above. Input is followed by an ADD or END directive as above.

The following subroutines are contained in this file

orbitl.f:1: SUBROUTINE ORBITL
orbitl.f:272: SUBROUTINE GAUSSL(N,X,W)
orbitl.f:323: SUBROUTINE SPLINE(N,X,Y,CM,ALPHA,BETA,B)
orbitl.f:494: SUBROUTINE TRIDIA(ALPHA,BETA,GAMMA,B,X,N)

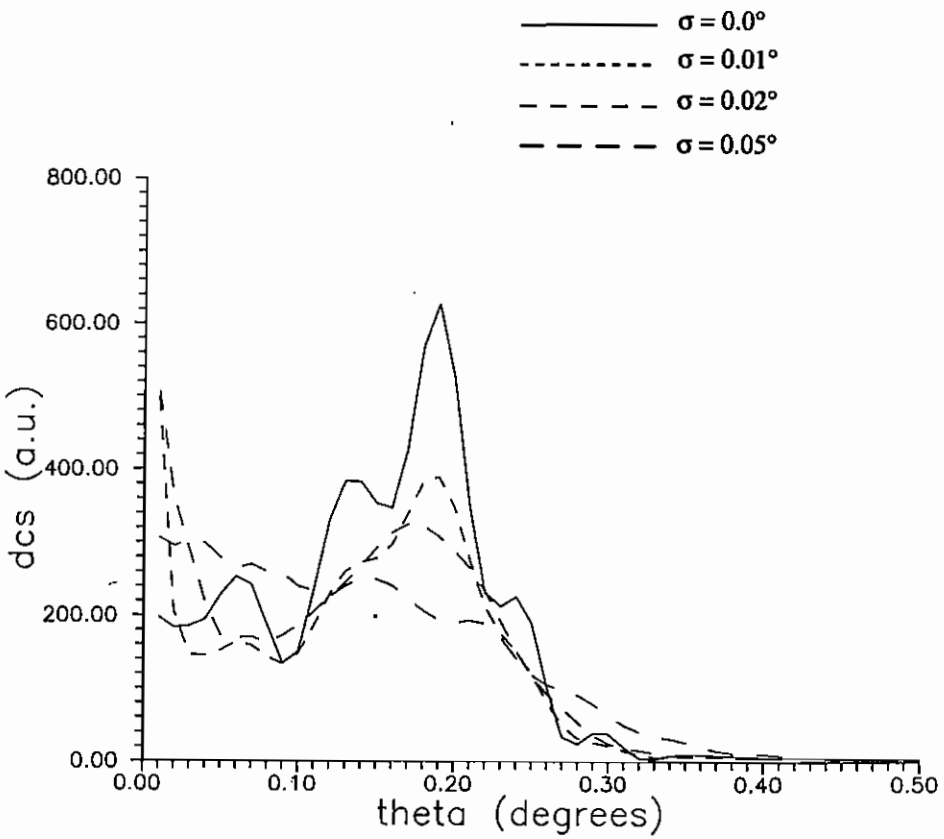


Fig.5

orbitl.f:508: SUBROUTINE SPLCOF(N,X,T)
orbitl.f:593: SUBROUTINE SPL(NDIM,E,CE,M1,M2,EP,NEP,J)
orbitl.srm.f:1: SUBROUTINE ORBITL

Example of input to orbitl [29, 30]:

```
1 orient
NOPR
TITLE
differential cross sections for Na – H+ 14–state, out of plane average
TITLE
 right to right transition, Na(3p) to H(2p)
ORIENT
CONVOLUTION
0.1
  300   10  5.0
90.0
 9 1.0
 9 1.0
 1 –0.707
 9 1.0
 9 1.0
 9 1.0
 9 1.0
 9 1.0
 5 –0.707
 6 0.707
 9 1.0
 9 1.0
 2          –0.707
 0
END
STOP
```

19 parameter.f

The following lines are the contents of file parameter.f, which contains the common block parameter definitions required for the compilation. Different runs may require recompilation with different parameters depending on the number of states, impact parameters, distances and angles to be treated and on the size of memory available in the computer. This file is the only one which need be edited, and the program can be remade after updating the time stamp on all the source code files required.

```
************************************************************************
* definition of parameters used in the EIKONXS program
```

```
************************************************************************
c number of distances in molecular data
      PARAMETER(IR=40)
c number of electronic states in the data
      parameter (IS=14)
      parameter (is2=2*is)
c number of coupling elements at each distance
      parameter (IC=(is*(is-1))/2)
c number of impact parameters for dcs
      parameter (iro=300)
c number of angles from dcs
      parameter (iang=161)
c thing used in diffxs
      parameter (idl=30001)
```

20 server.f

This program is a file server which communicates data from the Intel Concurrent File System to the host program using a special message protocol. It is used to handle the files TANGO.DAT and COPYS.DAT, which cannot be accessed directly, in conjunction with the set of routines cfsemul.f. See description of CFS by Moody [18], and details of routines and useage [16].

21 splinex.f

This is Antoine Salin's spline interpolation package [1] (with which we have had many pleasant and successful interpolations [sic]). I have extended it to interpolate over a vector of quantities and yield results, and first and second derivatives in the same routine call. Other routines are as in the later 'vectorised' version of the package.

The following subroutines are used

```
splinex.f:1:      SUBROUTINE SPLINE(N,X,Y,CM,ALPHA,BETA,B)
splinex.f:172:    SUBROUTINE TRIDIA(ALPHA,BETA,GAMMA,B,X,N)
splinex.f:186:    SUBROUTINE SPLCOF(N,X,T)
splinex.f:271:    SUBROUTINE INITIX(N,X,Y,C,CI)
splinex.f:287:    SUBROUTINE INTSPX(N,X,T,C1,K)
```



```
splinex.f:376:    SUBROUTINE INITIN(N,X,Y,CM,CI)
splinex.f:443:    SUBROUTINE SPL(NDIM,E,CE,M1,M2,EP,NEP,J)
splinex.f:452:    SUBROUTINE SPLP(NDIM,E,CE,M1,M2,EP,NEP,J)
splinex.f:462:    SUBROUTINE SPLP2(NDIM,E,CE,M1,M2,EP,NEP,J)
splinex.f:470:    SUBROUTINE SPLA(NDIM,E,CE,M1,M2,EP,EP1,EP2,NEP,J)
```

22 starks.f

This program developed with P.Salas [29] does the long-range part of the propagation analytically assuming the only residual coupling for $|z|>z_o$ is of the form w/R^2 (first-order Stark effect and rotational coupling). This is a special case, and a complete analysis of long-range effects is given by Salin [26]. The coupling constants and phases are computed numerically and the transition amplitudes, read from COPYS.DAT, are pre and post multiplied accordingly to yield $a^i_j(-\infty, +\infty)$ in the file TANGO.DAT.

Starting from equation

$$i\frac{\partial c}{\partial t} = Mc$$

write $c=A^{-1}b$ and assume $\frac{\partial A^{-1}}{\partial t}=0$ gives:

$$i\frac{\partial b}{\partial t} = AMA^{-1}b. \tag{70}$$

If A is the similarity transform matrix that diagonalises M then

$$\int_{t_0}^{t}\frac{\partial b}{b} = -i\int_{t_o}^{t} D\partial t \tag{71}$$

so $$b(t) = \exp\{-i\int_{t_0}^{t} D\partial t\}b(t_o) \tag{72}$$

Finally $$c(t) = A(t)\exp\{-i\int_{t_o}^{t} D\partial t\}A^{-1}(t_o)c(t_o). \tag{73}$$

Now in the long-range limit only coupling between degenerate states exists which is of the form $\rho w/R^2$ where w is a constant.

$$I = \int_{z_o}^{z} Ddz \simeq \rho w\int_{z_o}^{z}\frac{dz}{R^2} \tag{74}$$

where $R^2 = (\rho^2 + z^2)$

therefore $$I = \rho w\int_{z_o}^{z}\frac{\partial z'}{(\rho^2+z'^2)} = w[\tan^{-1}(\frac{z}{\rho}) - \tan^{-1}(\frac{z_o}{\rho})] \tag{75}$$

The asymptotic propagation is

$$c^m(-z_o) = A^m_n\exp\{-iD^n_n[-\tan^{-1}(\frac{z}{\rho_0})+\frac{\pi}{2}]A^{-1}{}^n_k c^k(-\infty)$$

$$c^m(\infty) = A^m_n \exp\{-iD^n_n[\tfrac{\pi}{2} - \tan^{-1}(\tfrac{z_o}{\rho})]A^{-1}{}^n_k c^k(z_o) \qquad (76)$$

A is the matrix which diagonalises the coupling at $z=z_o$, and both A and D are found numerically by the program. For correct numerical working of the diagonalisation small non–zero terms must be eliminated. The matrix M is of block–diagonal form (each block belonging to a manifold of degenérate states) and this may be assured by manipulating the vector MDE so that no coupling is computed between blocks *i.e.* setting values of MDE for those states to differ by an integer greater than one. The diagonalisation is then performed by automatically searching for the blocks using the routine MDIAGD.

As an example we observe the Stark effect in H(n=2) with states

$$|+\rangle = (|2s\sigma\rangle + |2p\sigma\rangle)/\sqrt{2}$$
$$|-\rangle = (|2s\sigma\rangle - |2p\sigma\rangle)/\sqrt{2} \qquad (77)$$

and $|\pi\rangle$. The splitting between $|+\rangle$ and $|-\rangle$ is $\Delta E = 6/R^2$ a.u. as $R\to\infty$. The coupling between these three states is asymptotically

$$M = \frac{1}{R^2}\begin{bmatrix} 0 & A & 0 \\ A & 0 & -iB \\ 0 & -B & 0 \end{bmatrix}$$; A = 3 — Stark effect, B = vb — local rotationwhich has eigenvalues $\Delta = \pm\sqrt{(A^2+B^2)}/R^2, 0$.

The transformation matrix is

$$S = \frac{1}{\sqrt{[2(A^2+B^2)]}}\begin{bmatrix} A & A & iB\sqrt{2} \\ \sqrt{(A^2+B^2)} & -\sqrt{(A^2+B^2)} & 0 \\ iB & iB & A\sqrt{2} \end{bmatrix} \qquad (78)$$

and the phase which is added in going from distance R_0 to ∞ is

$$\phi = \int_{R_0}^{\infty} \frac{\sqrt{(A^2+B^2)}}{R^2} dt \qquad \text{so that} \qquad (79)$$

$$U(t,t_0) = \begin{bmatrix} \frac{A^2}{C^2}\cos\phi + \frac{B^2}{C^2} & \frac{A}{C}i\sin\phi & \frac{AB}{C^2}\cos\phi - i\frac{AB}{C^2} \\ \frac{A}{C}i\sin\phi & \cos\phi & \frac{B}{C}\sin\phi \\ i\frac{AB}{C^2}\cos\phi - i\frac{AB}{C^2} & \frac{B}{C}\sin\phi & \frac{B^2}{C^2}\cos\phi + \frac{A^2}{C^2} \end{bmatrix} \qquad (80)$$

The following subroutines are contained in file starks.f

starks.f:1: SUBROUTINE STARKS
starks.f:133: SUBROUTINE RARR(NDIM,NS,T,A,N,NA)
starks.f:229: SUBROUTINE PERMUT(NDIM,NDIM2,NS,W,T,IPERM,FROW)
starks.f:255: SUBROUTINE SORT(NDIM,N,A,IPERM,FLAG)
starks.f:282: SUBROUTINE MDIAGD(NDIM,N,D,A,T,AT,FLAG)

Example of input to control program starks.f, some example output is shown in chapter II, The explicit breakdown of the diagonalisation into different symmetry blocks can be seen.

```
1 starks
NOPR
APROB
0.0
000
0.0
–0.18845 ...
–0.11155 ...
 0 2 2 5 0 3 6
STOP
```

23 tfft.f

This file contains Temperton's Multi–radix Fast Fourier Transform program [13]. It is used in evaluating emission spectra.

The following subroutines are called

tfft.f:1: SUBROUTINE FFTRIG(NFFT,TRIG)
tfft.f:11: SUBROUTINE FACTOR(IFAX,NFAC,N)
tfft.f:48: SUBROUTINE TFFT(A,C,N,TRIGS,IFAX,NFAC,ISKIP,ISIGN)
tfft.f:73: SUBROUTINE FFTB(A,B,C,D,N,TRIGS,IFAX,NFAC,ISKIP,ISIGN)
tfft.f:100: SUBROUTINE PASS(A,B,C,D,TRIGS,IFAC,LA,N,ISKIP,

24 total.f

This program evaluates the total cross section for transitions involving linear combinations of the amplitudes a^i_j from file TANGO.DAT using Simpson's–rule integration over the impact parameters. It allows a linear combination of amplitudes to be specified as in the other modules.

The following subroutines are contained in this file

total.f:1: SUBROUTINE TOTAL

total.f:139: SUBROUTINE SIMINT(N,X,Y,VAL)

Example of input on LU5 to control module total

```
1 total              cstype
NOPR
TITLE
****** Na-H+ 7 ESTADOS, rmax=30.  nro=47 ** -2*phi2, efin en h
TOTAL
999
 **** Transicion Na(3S) ---> H(2P) *****
  1  2  0.707
  1  3  -0.707
  0  0
STOP
```

Input lines for the total program take the same form as those to module orbitl, and describe linear combinations of states. An END directive is optional. Some example output from TOTAL is shown in chapter II.

25 yves.f

This program, on the nodes, sends the transition amplitude data and phases for each impact parameter to the host program to be stored in order in the file TANGO.DAT. On the host, it receives the data and writes it to the Concurrent File System of the Intel using the set of emulation routines cfsemul and file server program. In the sequential implementation it writes the binary data into file TANGO.DAT directly.

The following subroutines are contained in the file

yves.f:1: SUBROUTINE YVES(NS,C,PC,PHI,V,RO,ZPHAS,ZM)
yves.f:25: SUBROUTINE DUMP(IS,NS,ARA,FLAG)
yves.f:37: SUBROUTINE DUMPC(IS,NS,ARA,FLAG)

The storage format of the file TANGO.DAT is described in Appendix A.

26 ipsc.f

This file contains a number of dummy routines used in the sequential implementation to replace names of message-passing and special i/o routines present in the parallel version of EIKONXS on the Intel Hypercube. The makefile for the sequential version explicitly references this file, see Appendix B. These routines are:

```
subroutine csend(itype,n,nbytes,iaadr,ipid)
subroutine read(lu,nbytes,buffer)
subroutine un8(i,j,k,v1,v2)
subroutine un4(i,j,k,v1,v2)
subroutine cread(lu,buffer,nbytes)
subroutine cwrite(lu,buffer,nbytes)
subroutine copy8(i,j,k,v1,v2)
subroutine copy4(i,j,k,v1,v2)
subroutine crecv(itype,buffer,nbytes)
function mynode()
function numnodes()
function lseek(i,j,k)
```

IV Listing the molecular data.

A special program perubu.f has been written to manipulate the data. It can be used to list the data, or to put it into an ascii file for graphing of different quantities with an external package, e.g. on a PC or workstation.

This program is run sequentially, and interactively, apart from the EIKONXS package and is started with the command

perubu.out

It will ask for the name of an input file which can be output.txt, diab.dat or trijoc.dat. If you want to list data in perubu.dat it should be renamed "trijoc.dat" before use. Secondly you will be asked if the destination of the output is for eikonxs or for graphs. The former directive produces an output file which depends on the input

input	output
trijoc.dat	perubu.dat, output.txt
output.txt	perubu.dat
diab.dat	none

This permits data to be moved from one computer to another via the intermidiary of ascii file output.txt (a large file !).

The graphs directive produces an ascii listing in a file graph.txt which contains columns with firstly the internuclear distance R, then energies, or cr (coupling) out of any particular state, zdz (translation factor terms) or x2 (x^2 matrix element).

The following lines show a logged terminal session in which this program is run three times:

```
% ls
   perubu.out                                   trijoc.dat
% perubu.out
 *****PROGRAM.PERUBU*****
 directives: x2, cr, zdz, cr-zdz, energies, graphs, eikonxs

 name of input file
trijoc.dat
 37  7  0  0  0  0  0  1  1
 destination of output ?
eikonxs
   1  2
% ls
   output.txt                                   perubu.dat
   perubu.out                                   trijoc.dat
% rm perubu.dat
```



```
% perubu.out
 *****PROGRAM.PERUBU*****
 directives: x2, cr, zdz, cr-zdz, energies, graphs, eikonxs

 name of input file
output.txt
 37  7  0  0  0  0  0  1  1
 destination of output ?
eikonxs
   1  2
% ls
   output.txt                                   perubu.dat
   perubu.out                                   trijoc.dat
% perubu.out
 *****PROGRAM.PERUBU*****
 directives: x2, cr, zdz, cr-zdz, energies, graphs, eikonxs

 name of input file
trijoc.dat
 37  7  0  0  0  0  0  1  1
 destination of output ?
graphs
 which data ?
cr
 which initial state [i2] ?
 3
% ls
graph.txt                                       output.txt
   perubu.dat                                   perubu.out
   trijoc.dat
% tail graph.txt
  16.0      0.      0.      0.   6.365E-03  1.283E-02  0.480    -9.115E-02
  18.0      0.      0.      0.   4.948E-03  2.228E-03  0.509    -5.115E-02
  20.0      0.      0.      0.   2.415E-03 -5.813E-04  0.529    -2.558E-02
  22.0      0.      0.      0.   1.023E-03 -1.168E-03  0.545    -1.215E-02
  24.0      0.      0.      0.   4.192E-04 -1.223E-03  0.557    -5.626E-03
  26.0      0.      0.      0.   2.177E-04 -1.124E-03  0.568    -2.635E-03
  30.0      0.      0.      0.   1.540E-05 -8.386E-04  0.585    -5.592E-04
  40.0      0.      0.      0.  -3.230E-05 -7.803E-05  0.614     1.706E-05
  60.0      0.      0.      0.  -1.637E-06  3.691E-07  0.644    -3.133E-06
 100.       0.      0.      0.   1.192E-08 -1.351E-09  0.669     2.282E-11
%
```

Acknowledgements

The main work to develop this program was carried out under the auspices of the Deutsche ForschungsGemeinschaft SFB91 (1983–85) at the Universitaet Kaiserslautern, and the CCP6 (1985–87) at Daresbury Laboratory, and the Advanced Research Computing Group at Daresbury (1987 – present). Mariko Terao developed the emission program as part of her Ph.D. thesis at the Université Catholique de Louvain la Neuve, Belgium (1987).

We thank the Royal Society and CNRS for financial support enabling Dr. Christine Courbin to visit Daresbury from September 1987 to September 1988, and the Comunidad de Madrid for supporting Dr. Pedro Salas during his visits to Daresbury in 1987 and 1988. Work has also been partially supported under EEC CODEST contract numbers ST2J–0290–C(EDB) and ST2J–0033–1–UK and the Spanish D.G.I.C.y.T. contract number PS 87–0097.

This document has been prepared in support of EEC Science Plan contract number SC1*.0138.C(JR).

I thank Drs. A.Salin, C.Courbin, P.Salas, M.Terao and H.–J.Korsch for their help in developing this package between 1985 and 1989 and for their support and helpful suggestions. In addition I particularly thank Dr.A.S.Dickinson and Prof.R.McCarroll for introducing me to charge–transfer theory.

References

[1] C.Gaussorgues, R.D.Piacentini and A.Salin "Multistate molecular treatment of atomic collisions in the impact parameter approximation. I – integration of coupled equations and calculation of transition amplitudes for the straight–line case" Comp. Phys. Comm. 10 (1975) 223–233

[2] R.D.Piacentini and A.Salin "Multistate molecular treatment of atomic collisions in the impact parameter approximation. II – calculations of differential cross sections from the transition amplitudes for the straight–line case" Comp. Phys. Comm. 13 (1977) 57–62

[3] R.D.Piacentini and A.Salin "Multistate molecular treatment of atomic collisions in the impact parameter approximation. III – integration of coupled equations and calculation of transition amplitudes for Coulomb trajectories" Comp. Phys. Comm. 12 (1976) 199–203

[4] G.D.Billing and M.Baer "A propagator method for integration of classical–trajectory equations " Chem. Phys. Lett 48 (1977) 372–6

[5] M.Gargaud 1980 These de 3e cycle de l'universite de Bordeaux

[6] Burlisch and Stoer Numerische Mathematik

[7] M.R.Flannery and J.K.McCann J. Chem. Phys 63 (1975) 4695
M.R.Flannery and J.K.McCann J. Chem. Phys 69 (1978) 5275

[8] A.E. dePristo J. Chem. Phys. 78 (1983) 1237

[9] F.Wolf, R.J.Allan and H.J.Korsch "A critical look at quasiclassical trajectory methods in inelastic collisions" Comm. At. Mol. Phys. 18 (1986) 107–23

[10] R.J.Allan, A.Bähring and J.Hanssen J. Phys. B 18 (1985) 1999–2019

[11] F.T.Smith "Diabatic and Adiabatic representations for atomic collision problems" Phys. Rev. A 179 (1969) 111–23

[12] M.Terao "L'observation des spectres émis au cours d'une collision constitue une méthode directe pour la determination des populations des orbitales moléculaires actives" Thèse Annexe, Universite Catholique de Louvain la Neuve, Belgium (1987)

[13] C.Temperton "Self–sorting Mixed–radix Fast Fourier Transforms" J. of Comp. Phys. 52 (1983) 1–23

[14] J.S.Briggs and K.Dettman "Theory of continuum emission during ion–atom collisions" J. Phys. B 10 (1977) 1113–32

[15] M.C.Pacher "Emision de radiacion durante collisiones atomicas de baja energia" Tesis de Licentura en Ciensas Fisicas, Universidad Buenos Aires (1985)

[16] R.J.Allan "FORTRAN–77 Programming of Parallel Computers" D.L. Technical Memorandum DL/SA/TM61T

[17] M.Terao, C.Harel, A.Salin and R.J.Allan "Theoretical study of singleelectron capture in $He^{2+} - H^-$ collisions" Z. Phys. D 7 (1988) 319–32

[18] D.Moody "Intel ins and outs" Parallelogram 15 (1989)

[19] R.T.Pack "Atom–diatomic molecule scattering" J. Chem. Phys. 60 (1974) 633–9

[20] L.F.Errea, L.Mendez and A.Riera J. Phys. B 15 (1982) 101–10

[21] A.Riera Phys. Rev. A 30 (1984) 2304 and L.F.Errea, J.M.Gomez–Llorente, L.Mendez and A.Riera "Practical Criterion for the determination of translation factors. II Application to He^{2+}+H(1s) collisions" Phys. Rev. A 32 (1985) 2158–65

[22] R.J.Allan, C.Courbin, P.Salas and P.Wahnon "State–selective effects in the differential cross section for electron capture from laser–excited sodium atoms by protons" J. Phys. B. Letters (1990) in press

[23] R.J.Allan "Documentation of the QUANTXS program" Daresbury Laboratory Technical Memorandum (1990) in preparation

[24] H.Hellmann "Einfuerung in die Quantentheorie" (Deuticke, Leipzig, 1937) pp 285–86 and R.P.Feynman Phys. Rev. 56 (1939) 340

[25] V.Sidis "Simple Expression for the off–diagonal matrix elements of the d/dR operator between electronic states of a diatomic molecule" J. Chem. Phys. 55 (1971) 5838–9

[26] A. Salin Journal de Physique 45 (1984) 671–80

[27] M.Terao, R.J.Allan and H.Bachau "Theoretical study of two–electron capture in $He^{2+}H^-$ collisions" Daresbury Laboratory Annual Report (1990) and 1st ATMOP Conference, Belfast, (1990)

[28] D.W.Peacemann and H.H.Rachford Jr. "The numerical solution of parabolic and elliptic differential equations" J. Soc. Indust. Appl. Maths. 3 (1955) 28–41

[29] C.Courbin, R.J.Allan, P.Salas and P.Wahnón "Total and differential charge–transfer cross sections in H^+ + Na(3s) or Na^*(3p) collisions" J. Phys. B (1990) in press

[30] R.J.Allan, C.Courbin, P.Salas and P.Wahnón "State selective effects in collisional electron capture from non–isotropic targets" J. Phys. B (1990) in preparation

[31] A.S.Dickinson J. Phys B 14 (1981) 3685–91

[32] VOGLE and VOPL are available from Eric H. Echidna, Software Support

Programmer, Department of Engineering Computer Resources, Faculty of Engineering, University of Melbourne, Victoria 3052, Australia, echidna@munnari.OZ.AU

Appendix A File structure and default names

In this appendix we describe the standard files accessed by the program, and in the case of binary data files, the format for storage of quantities in them so that the user may interface his own programs to them. Only the first file is necessary as input, the others being created from it by the various program modules. Two formats are available for the input file. Two output files contain the transition amplitudes and the time–dependent propagator matrix and result from running aprobe or dprobe. These files are used by other modules to compute cross sections and related quantities.

LU1 – PERUBU.DAT adiabatic molecular (energies and couplings) input
LU2 – TANGO.DAT partial cross section (transition amplitudes and phases) output
LU3 – DIFFXS.DAT differential cross section output
LU4 – DUMP.DAT dump file for intermediate output
LU5 – STDIN directives and run–time data
LU6 – STDOUT output listing (optional with NOPRINT)
LU8 – DIAB.DAT diabatic potential data
LU9 – COPYS.DAT workfile, copy of tango.dat for same or different symmetry
LU10 – EVOLU.DAT incoming evolution propagator
LU11 – ADIP.DAT adiabatic dipole moment data for radiative collisions
LU12 – DDIP.DAT diabatic dipole moment data

PROBn.DAT see section 18 chapter III. Used to store state–to–state probabilities. One file is opened for each initial and final condition, if the program is run again for a different velocity the files are appended so that 3–dimensional surfaces of P(b,v) are collected.

The file PERUBU.DAT holds the basic adiabatic molecule energies and matrix elements essential to the program (their form is given in equation (28)). It is of course possible to directly introduce diabatic quantities in the file DIAB.DAT (see below) using an ad hoc formulation, but we have already warned of the dangers of doing so [17]. Assuming that adiabatic quantities can be calculated using either finite–difference, analytic derivatives or other formulae for d/dR such as the Hellman–Feynman or Sidis approximations [24,25], then they must be stored in binary as indicated in the following code fragment. The file contains a number of concatenated blocks of data, each starting with a line nr, ns, me: number of distances in the block, number of states and parity of the states. Each block is read and then the distances are sorted as shown in the example

output in chapter II.

```
      ...
      COMMON/VALEN/R(IR),E(IR,IS),CE(IR,IS),E2(IR,IS),CE2(IR,IS),
     1    NE,NPOS
      COMMON/COUPLE/BRR(IR),TR(IR,IC),CTR(IR,IC),
     1    TR2(IR,IC),CTR2(IR,IC),NC,NR
      ...
      NRI=1
   42 READ(1,END=43) NR,NS,(ME(I),I=1,NS)
      NC=NS*(NS-1)/2
      NE=NR
      NK=0
      DO 10 K=1,NS
      DO 20 KK=K,NS
      IF(K.NE.KK) NK=NK+1
      IF(IABS(ME(K)-ME(KK)).GT.1) GO TO 21
      READ(1)(R(I),CR(I),ZDZ(I),X2(I),E(I,K),
     1       E(I,KK),I=NRI,NR+NRI-1)
      DO 30 I=NRI,NR+NRI-1
      RR=R(I)
      CRAD=CR(I)
      C(1)=ZDZ(I)
      C(2)=X2(I)
   33   BRR(I)=RR
      IF(K.EQ.KK) GO TO 19
      E(I,K)=-E(I,K)
      E(I,KK)=-E(I,KK)
      IF(ME(K).NE.ME(KK)) GO TO 18
      TR(I,NK)=CRAD+C(1)*RR/(RR*RR+ALPHA*ALPHA)
      TR2(I,NK)=C(2)*1.5/(RR*RR)
      GO TO 30
   19 CONTINUE
      E(I,K)=-E(I,K)
      E2(I,K)=C(2)*1.5/(RR*RR)
      GO TO 30
   18 CONTINUE
      TR(I,NK)=CRAD-C(1)
      TR2(I,NK)=0.0
   30 CONTINUE
      GO TO 20
   21 CONTINUE
      DO 31 I=1,NR
      TR2(I,NK)=0.0
   31 TR(I,NK)=0.0
   20 CONTINUE
   10 CONTINUE
      NRI=NRI+NR
      GOTO 42
   43   NRI=NRI-1
      ...
```

file is of a different format, and must contain lifetimes of the states for the non–unitary calculation which will follow. Flux is usually lost to ionisation of some form so that the lifetimes will account for implicit coupling to the unbound continuum. The code fragment describing this new format is taken from program gamma.f. This will be eventually made user–defined with a format descriptor file.

```
   ...
c note different format for storage in this file, should really be given a
c different name to avoid confusion ! However I've defined a new directive
c 'NEWFORMAT' which sets iprog to between 11 and 20 for this. Old format
c should have iprog between 1 and 10
c files with the non–diagonal lifetimes will have between 21 and 30.
      READ(1) NR,NS,(ME(I),I=1,NS)
      if(iprog.gt.10.and.iprog.lt.20)then
      itype=10
      if(inode.eq.0)call csend(itype,ns,4,ihost,0)
      READ(5,'(F12.5)')ALPHA
      if(inode.eq.0)write(6,81)alpha
 81    format(1x,' alpha = ',f10.5)
      NC=NS*(NS–1)/2
      NE=NR
      read(1)(r(i),i=1,nr)
c diagonal parts energy and lifetime (both real) e–i*dlife
      do 15 k=1,ns
 15      read(1)(e(i,k),dlife(i,k),i=1,nr)
      NK=0
      DO 10 K=1,NS
      DO 20 KK=K,NS
        if(k.eq.kk) go to 25
        NK=NK+1
        IF(IABS(ME(K)–ME(KK)).GT.1) GO TO 21
c off diagonal couplings for radial or rotational,
c tr is combined coupling and etf element, tr2 is x2 element
        READ(1)(tr(I,nk),tr2(I,nk),I=1,NR)
        goto 20
 21      do 22 i=1,nr
           tr(i,nk)=0.0
           tr2(i,nk)=0.0
 22      continue
 23      continue
      go to 20
c diagonal part of x2 matrix
 25   read(1)(e2(i,k),i=1,nr)
   20 CONTINUE
   10 CONTINUE
      else
        write(6,'(" ***** error gamma ***** format ")')
      end if
     ...
```


produced on LU4=DIAB.DAT by calling the diab.f program. Alternatively an ad–hoc method may have been employed to create the data which should be formatted as follows

```
     ...
      WRITE(6,80)
  80  FORMAT(' *****PROGRAM DIAB*****')
      READ(5,'(F12.5)')ALPHA
      nri=1
c read multiple files and sort them
  42  READ(1,end=43) NR,NS,(ME(I),I=1,NS)
      write(6,'(" reading nr, ns = ",2i3)')nr,ns
      NC=NS*(NS–1)/2
      NE=NR
      NK=0
      DO 10 K=1,NS
      DO 20 KK=K,NS
      IF(K.NE.KK) NK=NK+1
      IF(IABS(ME(K)–ME(KK)).GT.1) GO TO 21
      READ(1)(R(I),CR(I),ZDZ(I),X2(I),E(I,K),
     1       E(I,KK),I=nri,NR+nri–1)
      DO 30 I=nri,Nr+nri–1
      RR=R(I)
      CRAD=CR(I)
      C(1)=ZDZ(I)
      C(2)=X2(I)
  33  BRR(I)=RR
      IF(K.EQ.KK) GO TO 19
c make energies negative
      E(I,K)=–E(I,K)
      E(I,KK)=–E(I,KK)
      IF(ME(K).NE.ME(KK)) GO TO 18
      TR(I,NK)=CRAD+C(1)*RR/(RR*RR+ALPHA*ALPHA)
      TR2(I,NK)=C(2)*1.5/(RR*RR)
      GO TO 30
   19 CONTINUE
      E(I,K)=–E(I,K)
      E2(I,K)=C(2)*1.5/(RR*RR)
      GO TO 30
   18 CONTINUE
      TR(I,NK)=CRAD–C(1)
      TR2(I,NK)=0.
   30 CONTINUE
      GO TO 20
   21 CONTINUE
      DO 31 I=nri,NR+nri–1
      TR2(I,NK)=0.
   31 TR(I,NK)=0.
   20 CONTINUE
   10 CONTINUE
      nri=nri+nr
      goto 42
```

```
 43   nr=nri-1
      ne=nr
      ...
```

Output from the proopagation stage of the program is stored in two possible files. One of them is TANGO.DAT on LU=2 and is always used, it contains the transition amplitudes, the other one is EVOLU.DAT on LU=10 and is only used when directed to store the evolution matrix. This is of interest in radiative transitions and to study the time-evolution of the wavefunction. This data can currently only be produced by running the dprobe module. The data structure of these two files is briefly described by the following lines of code:

Storage of data in TANGO.DAT

```
      WRITE(2)NV,NRO,NS,RPHAS
repeat for each impact parameter
      DO 100 L=1,NRO
 100  WRITE(2)V,RO(L),(PHI(I,L),I=1,NS),((C(J,K,L),J=1,NS),K=1,NS)
```

Storage of data in EVOLU.DAT

```
first line of file
      WRITE(10)SYMBOL,V,RHO,NS,ZIN
repeat for each impact parameter
      do 100 l=1,nro
do half propagation in dprobe for symmetric trajectory
      do 110 t=-t0,0,tstep
      ...
at each time step, store propagator
      WRITE(10)SYMBOL,Z1,((EV(I,J),I=1,NS),J=1,NS)
 110  continue
at end of calculation, initial and final amplitudes
      WRITE(10)SYMBL2,ZMAX,((C(J,I),J=1,NS),I=1,NS)
      WRITE(10)SYMBL3,ZINT,((CCONJG(J,I),J=1,NS),I=1,NS)
 100  continue
```


Appendix B

Makefile for Intel iPSC/2

```
#########################################################################
#                                                                       #
#          Makefile for EIKONXS program  for iPSC/2                     #
#                                                                       #
#########################################################################
#
#########################################################################
#     Hostlib  creates and maintains 1 library:                         #
#          eikonxs.a                                                    #
#     and builds eikonxs                                                #
#     with the main programs main.f (cube) and main0.f (host)           #
#          the Executable eikonxs0.out is built up.                     #
#########################################################################


SOURCE=         \
          aprobe.f\
          diag.f\
          matrix.f\
          hamilton.f\
          splinex.f\
          starks.f\
          yves.f\
          diffxs.f\
          diab.f\
          dprobe.f\
          cfsemul.f

FFLAGS1= -sx

LIB=          eikonxs.a

all:          eikonxs0.out $(PROG) server.out
          @echo eikonxs all finished

$(LIB):       Stamp1
          ar rv $(LIB) *.o
          rm *.o

Stamp1:       $(SOURCE)
          f77 -c $(FFLAGS1) $?
          touch Stamp1

eikonxs.out:    main.f $(LIB)
          $(F77) $(FFLAGS1) -o $(PROG) main.f $(LIB) -node
          @echo eikonxs link completed
```

```
        rm *.o

eikonxs0.out:   main0.f orbitl.f cfsemul.f
        f77 -o eikonxs0.out main0.f orbitl.f cfsemul.f -host
        rm *.o

server.out:     server.f
        f77 -o server.out -sx server.f -node
        rm *.o
```

Makefile for Convex C220

```
LFLAGS= -O2

CFLAGS= -O2 -a1

GRAPHLIB= /mnt1/ge/naggraf/naggraf.a -lnag -lghost -lgridplus -lselanar

SOURCE =    main.f   aprobe.f  \
 diab.f          dprobe.f  \
 diag.f          matrix.f  \
 hamilton.f        orbitl.f  \
 copys.f         diffxs.f  \
 total.f        splinex.f \
 starks.f        yves.f    \
 defln.f         graphics.f\
 ipsc.f

OBJECTS =   main.o   aprobe.o  \
 diab.o          dprobe.o  \
 diag.o          matrix.o  \
 hamilton.o        orbitl.o  \
 copys.o         diffxs.o  \
 total.o        splinex.o \
 starks.o        yves.o    \
 defln.o         graphics.o\
 ipsc.o

FC = fc $(CFLAGS)

PROGRAM = /priv3/rja/eikonxs/eikonxs.out

.KEEP_STATE:
$(PROGRAM):   $(OBJECTS)
  $(FC) $(OBJECTS) -o $(PROGRAM) $(GRAPHLIB)
  @echo link all finished

.f.c: ; $(FC) -c $*.f
```

This form of makefile is also appropriate to compilation on other UNIX processors, and in particular to the Alliant FX2800. The compiler options must of course be changed appropriately.

The following files are different in the distributed memory and shared memory versions:

yves.f, ipsc.f, cfsemul.f, main0.f, main.f,

They contain explicit reference to the message-passing utilities and concurrent i/o system.

Appendix C Optional partial linking of routines

a) calculation of transition amplitudes from adiabatic data:
APROBE, DIAG, HAMILTON, MAIN, MATRIX, SPLINEX, YVES, DEFLN
files LU1, LU2, LU5, LU6 (LU10, LU11 optional)

b) diabatisation of adiabatic data for use by either dprobe or the quantal program QUANTXS
DIAB, DIAG, MAIN, MATRIX, SPLINEX, YVES
files LU1, LU5, LU6, LU8 (LU11, LU12 optional)

c) calculation of transition amplitudes from diabatic data
APROBE, DPROBE, DIAG, HAMILTON, MAIN, MATRIX, SPLINEX, YVES
files LU2, LU5, LU6, LU8 (LU11, LU12 optional)

d) copy of work files containing transition amplitudes
MAIN, COPYS
files LU2, LU5, LU6, LU9

e) propagation of long–range coupling
MAIN, STARKS
files LU2, LU5, LU6, LU9

f) calculation of total cross sections
MAIN, TOTAL
files LU2, LU5, LU6, LU9

g) calculation of differential cross sections
MAIN, DIFFXS
files LU2, LU3, LU5, LU6, LU9

h) calculation of orientation parameters
MAIN, ORBITL
files LU2, LU5, LU6, LU9

i) calculation of radiative transition spectrum
MAIN, EMISSION, TFFT, ...
files LU2, LU5, LU6, LU9, LU11, LU12