JMATRIX – PACKAGE FOR RELATIVISTIC J-MATRIX CALCULATIONS IN ELASTIC SCATTERING OF ELECTRONS FROM MODEL POTENTIALS

PAWEŁ SYTY AND JÓZEF E. SIENKIEWICZ

Department of Theoretical Physics and Quantum Information
Gdańsk University of Technology
Narutowicza 11/12, 80-233 Gdańsk, Poland

(received: 18 November 2016; revised: 5 December 2016;
accepted: 12 December 2016; published online: 9 January 2017)

Abstract: We present a software package JMATRIX[1], consisting of two computer codes written in FORTRAN 95 and parallelized with OpenMP, implementing the so-called J-matrix method, applied to elastic scattering of electrons on the radial potential, vanishing faster than the Coulomb potential. In the J-matrix method, the physical scattering problem is replaced by using a well-defined model which is solved analytically. The presented software implements both non-relativistic and relativistic versions of the method, and allows calculations of scattering phase shifts as well as cross sections, in cases when the scattering potential is given through an analytical formula.

We performed test calculations for the scattering potential modeled as a truncated Coulomb potential. We show that the numerical phase shifts to converge as we increase the size of the basis used to truncate the scattering potential, and that the method is suitable for calculating the total differential momentum transfer and spin polarization cross sections, using the partial-wave analysis.

Keywords: J-matrix method, relativistic, electron elastic scattering, phase shifts, Dirac partial-waves analysis, differential, total, momentum transfer, spin polarization cross sections

DOI: https://doi.org/10.17466/tq2017/21.1/c

1. Introduction

The J-matrix method is an algebraic method in the quantum scattering theory. A non-relativistic version of the method was introduced in 1974 by Heller and Yamani [1, 2] and developed by Yamani and Fishman [3] a year after. The relativistic version was introduced by Horodecki [4] and the extended version by

1. Available from: http://jmatrix.sylas.info/

Alhaidari *et al.* [5]. The method is based on the fact that the radial kinetic energy operator is tridiagonal in some suitable basis (such as the Gaussian, Laguerre or oscillator basis set). The scattering potential (vanishing faster than the Coulomb potential) is truncated in N elements of the selected basis. Then, using some algebraic methods, one can find a formula for the tangent of the approximated phase shift ($\tan\delta_N$). We expect that for $N \to \infty$, this approximate value converges to the exact value, $\tan\delta$.

The main goal of the present work was to implement and illustrate the application of the relativistic version of the J-matrix method, but the non-relativistic version was implemented, as well. The very first calculations using the early version of the program were presented in [6], but since that time the code has been significantly improved. In the current version, scattering potentials can be easily modeled as the square-well potential, the truncated Coulomb potential, the Yukawa potential, or may be given by any analytical formula. At the present time, the program calculates phase shifts for different angular quantum numbers and a range of energies. Then, a standard partial-wave analysis is used to obtain cross sections.

The package JMATRIX is split into two separate codes, sharing some numerical libraries. The first code, called JMATRIX-CONV, allows tracking the convergence of numerical phase shifts, while increasing the basis size. It may be useful to find the optimal basis size for a specific system, as a compromise between accuracy and computational time. The maximum basis size, N, is limited to 5000.

The second code, JMATRIX-CS, calculates the total differential momentum transfer and spin-polarization cross-sections using the partial waves analysis. Specific properties of the J-matrix method allow calculating phase shifts for many projectile energies with relatively small computational time. This feature was utilized in both codes. As the computational complexity is proportional to N^4, the calculations for $N = 1000$ and higher may take several hours, especially in the relativistic case.

The package is distributed as open source and is freely available at http://jmatrix.sylas.info as a gzipped tar or 7-zip file.

2. Theoretical method

Our task is to find an approximate solution of the scattering problem on the radial potential $V = V(r)$ vanishing in infinity faster than the Coulomb potential. In this section we give a rather short review of the J-matrix theory of scattering only, nevertheless it should be sufficient for understanding the main idea of the method. A detailed description of the method can be found in [1–7]. We use atomic units in all equations.

2.1. Relativistic J-matrix method

Let us start with the Dirac equation:

$$\left(H_0-\frac{E}{c\hbar}+\frac{V}{c\hbar}\right)\Psi(r)\equiv\begin{pmatrix}\frac{mc^2-E+V}{c\hbar} & -\frac{d}{dr}+\frac{\kappa}{r}\\ \frac{d}{dr}+\frac{\kappa}{r} & \frac{-mc^2-E+V}{c\hbar}\end{pmatrix}\Psi(r)=\begin{pmatrix}0\\0\end{pmatrix} \tag{1}$$

where $V=V(r)$ is the scattering potential mentioned before, E is the total energy of the projectile. Here, $\Psi=\Psi(r)$ is the relativistic wave function, satisfying the following asymptotic condition:

$$\Psi(r)\xrightarrow{r\to\infty}\sin\left(kr-\frac{\pi l}{2}\right)+\tan\delta\cos\left(kr-\frac{\pi l}{2}\right) \tag{2}$$

where δ is the relativistic phase shift, l is the angular momentum of the projectile, $\kappa=-l-1$ or $\kappa=l$, and $k\equiv\sqrt{(E-mc^2)(E+mc^2)}/c\hbar$.

Let us choose the basis set [4]:

$$\Phi_n^+(r)\equiv\begin{pmatrix}\phi_n^l(\lambda r)\\0\end{pmatrix} \tag{3}$$

$$\Phi_n^-(r)\equiv\begin{pmatrix}0\\\psi_n^l(\lambda r)\end{pmatrix},\quad \psi_n^l(\lambda r)=\left(\frac{\kappa}{r}+\frac{d}{dr}\right)\phi_n^l(\lambda r) \tag{4}$$

The most popular choice of basis functions $\left\{\phi_n^l\right\}$ is the complete oscillatory basis set. The other popular basis sets are the Laguerre and Gaussian bases. For the latter cases, functions ϕ_n^l are biorthonormal, so that $\left\langle\overline{\phi}_m^l\middle|\phi_n^l\right\rangle\equiv\int_0^\infty\overline{\phi}_m^l(\lambda r)\phi_n^l(\lambda r)dr=\delta_{mn}$. Elements of oscillatory, Laguerre and Gaussian basis sets are collected in Table 1.

In such defined bases, the term

$$J_{mn}^{ss'}\equiv\left\langle\Phi_m^s\middle|\left(H_0-\frac{E}{c\hbar}\right)\Phi_n^s\right\rangle,\quad s,s'=\pm,\quad m,n=0,1,... \tag{5}$$

gives the tridiagonal form (Jacobi matrix). These so called J-matrix elements $J_{mn}^{ss'}$ can be written as

$$J_{mn}=\begin{pmatrix}-k\epsilon\left\langle\phi_m^l\middle|\phi_n^l\right\rangle & \left\langle\psi_m^l\middle|\psi_n^l\right\rangle\\ \left\langle\psi_m^l\middle|\psi_n^l\right\rangle & -\frac{k}{\epsilon}\left\langle\psi_m^l\middle|\psi_n^l\right\rangle\end{pmatrix}_{ss'} \tag{6}$$

where $\epsilon\equiv\sqrt{\frac{E-mc^2}{E+mc^2}}$. Now, the J-matrix elements can be easily calculated (see [4] for details).

To ensure proper asymptotic behavior, we introduce sine- and cosine-like solutions of equation:

$$\left(H_0-\frac{E}{c\hbar}\right)\Psi_U=\begin{pmatrix}\Omega_U\overline{\phi}_0^l\\0\end{pmatrix},\quad \Omega_S=0,\quad \Omega_C=-\frac{\epsilon}{s_0^l} \tag{7}$$

Table 1. Elements of Laguerre, Gaussian (biorthonormal) and oscillator (complete) basis set and elements of expansion of sine-like and cosine-like solutions; $L_n^{(\alpha)}$ and $C_n^{(\alpha)}$ are Laguerre and Gegenbauer polynomials, respectively; ${}_2F_1$ and ${}_1F_1$ are hypergeometric functions, $\lambda>0$ is a scaling parameter ($\lambda\neq 0.5$)

Laguerre basis	
ϕ_n^l	$(\lambda r)^{l+1}\exp\left(-\frac{\lambda r}{2}\right)L_n^{(2l+1)}(\lambda r)$
$\bar{\phi}_n^l$	$\frac{n!}{\lambda^2\Gamma(n+2l+2)}\frac{1}{r}\phi_n^l(\lambda r)$
s_n^l	$\frac{2^l\Gamma(l+1)n!(\sin\theta)^{l+1}}{\Gamma(n+2l+2)}C_n^{(l+1)}(\cos\theta)$
c_n^l	$\frac{-2^l\Gamma(l+\frac{1}{2})n!}{\sqrt{\pi}\Gamma(n+2l+2)(\sin\theta)^l}\,{}_2F_1\left(-n-2l-1,n+1,\frac{1}{2}-l;\ \sin^2\left(\frac{\theta}{2}\right)\right),\quad \sin\theta\equiv\frac{k\lambda^{-1}}{k^2\lambda^{-2}+\frac{1}{4}}$

Gaussian basis	
ϕ_n^l	$(\lambda r)^{l+1}\exp\left(-\frac{\lambda^2r^2}{2}\right)L_n^{(l+\frac{1}{2})}(\lambda^2r^2)$
$\bar{\phi}_n^l$	$\frac{2n!}{\lambda^2\Gamma(n+l+\frac{3}{2})}\phi_n^l(\lambda r)$
s_n^l	$\frac{\sqrt{2\pi}n!(-1)^n}{\Gamma(n+l+\frac{3}{2})}\exp\left(-\frac{\eta^2}{2}\right)\eta^{l+1}L_n^{(l+\frac{1}{2})}(\eta^2)$
c_n^l	$\frac{\sqrt{\frac{2}{\pi}}\Gamma\left(l+\frac{1}{2}\right)(-1)^n n!}{\Gamma(n+l+\frac{3}{2})}\exp\left(-\frac{\eta^2}{2}\right)\eta^{-l}\,{}_1F_1\left(-n-l-\frac{1}{2},\frac{1}{2}-l;\ \eta^2\right),\quad \eta\equiv\frac{k}{\lambda}$

Oscillator basis	
ϕ_n^l	$r(-1)^n\sqrt{\frac{2n!\lambda^3}{\Gamma(n+l+3/2)}}(\lambda r)^l\exp\left(-\frac{\lambda^2r^2}{2}\right)L_n^{(l+1/2)}\left(\lambda^2r^2\right)$
s_n^l	$\sqrt{\frac{\pi n!}{\lambda k\Gamma(n+l+\frac{3}{2})}}\left(\frac{k}{\lambda}\right)^{l+1}\exp\left(-\frac{k^2}{2\lambda^2}\right)L_n^{(l+\frac{1}{2})}\left(\frac{k^2}{\lambda^2}\right)$
c_n^l	$\frac{-1^l}{\Gamma(-l+\frac{1}{2})}\sqrt{\frac{\pi n!}{\lambda k\Gamma(n+l+\frac{3}{2})}}\left(\frac{k}{\lambda}\right)^{-l}\exp\left(-\frac{k^2}{2\lambda^2}\right)\,{}_1H_1\left(-n-l-\frac{1}{2},\frac{1}{2}-l,\frac{k^2}{\lambda^2}\right)$

In the above, the indices U,C correspond to sine- and cosine-like solutions. The solutions, expanded in the selected basis, are as follows:

$$U(k,r)=\sum_{n=0}^{\infty}u_n^l(k)\begin{pmatrix}\phi_n^l\\ \frac{\epsilon}{\kappa}\psi_n^l\end{pmatrix},\quad U=S,C,\quad u=s,c \tag{8}$$

Now, the following recursive relations are fulfilled:

$$\sum_{n=0}^{\infty}J_{mn}s_n^l=0 \tag{9}$$

$$\sum_{n=0}^{\infty}J_{mn}c_n^l=-\frac{k}{2s_0^l}\bar{\varphi}_0^l \tag{10}$$

$$J_{00}s_0^l+J_{01}s_1^l=0,\quad J_{00}c_0^l+J_{01}c_1^l=-\frac{k}{2s_0^l} \tag{11}$$

$$J_{n,n-1}u_{n-1}^l+J_{n,n}u_n^l+J_{n,n+1}u_{n+1}^l=0;\quad u=s,c;\quad n>1 \tag{12}$$

The elements s_n^l and c_n^l can be found using the above recursive relations. The explicit forms of these coefficients are collected in Table 1.

Let us now replace this scattering potential by a truncated potential operator:

$$V^N_{mn} = \begin{pmatrix} \left\langle \phi^l_m \,\middle|\, V/c\hbar\phi^l_n \right\rangle & 0 \\ 0 & \left\langle \psi^l_m \,\middle|\, V/c\hbar\psi^l_n \right\rangle \end{pmatrix}, \quad m,n=0,1,...,N-1 \tag{13}$$

Hence, the scattering equation has the following form:

$$\left(H_0 - \frac{E}{c\hbar} + V^N\right)\Psi^N(r) \equiv \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{14}$$

with the expanded solution:

$$\Psi^N = \sum_{m=0}^{N-1} \begin{pmatrix} d^+_m \phi^l_m \\ d^-_m \frac{\epsilon}{k} \psi^l_m \end{pmatrix} + \sum_{m=N}^{\infty} \begin{pmatrix} (s^+_m + tg\delta_N c^+_m)\,\phi^l_m \\ (s^-_m + tg\delta_N c^-_m)\,\psi^l_m \end{pmatrix}. \tag{15}$$

Equations for $m > N$ are automatically fulfilled. Solving the remaining equations, one can find the tangent to the approximated phase shift:

$$\tan\delta_{N,l} = -\frac{s^l_{N-1}(k) + \frac{2\epsilon}{k} G^{++}_{N-1,N-1}(E)\,\mathrm{J}_{N,N-1}(k)\,s^l_N(k)}{c^l_{N-1}(k) + \frac{2\epsilon}{k} G^{++}_{N-1,N-1}(E)\,\mathrm{J}_{N,N-1}(k)\,c^l_N(k)} \tag{16}$$

where $G^{ss'}_{mn}(E) = \sum\limits_{p=\pm}\sum\limits_{i=0}^{N-1} c\hbar \frac{\Gamma^{sp}_{mi}\Gamma^{s'p}_{ni}}{E^p_i - E}$, and $\left(\Gamma^\dagger P^\dagger_N \left(H_0 + \frac{V}{c\hbar} - \frac{E}{c\hbar}\right) P_N \Gamma\right)^{ss'}_{mn} = \frac{1}{c\hbar}\left(E^s_n - E\right)\delta_{nm}\delta_{ss'}$. Matrix G can be viewed as the matrix approximating the Green function.

For $N \to \infty$, what is connected with a reduction in the inaccuracy in approximating the scattering potential, $\tan\delta_{N,l}$ should converge to the exact value $\tan\delta$, and simultaneously, approximate $\delta_{N,l}$ should approach the exact scattering phase, δ.

2.2. The non-relativistic J-matrix method

In the non-relativistic version of the J-matrix method we start from the Schrodinger equation, by analogy to the relativistic case. Then, we replace the scattering potential by a truncated potential operator:

$$V^N = P^\dagger_N V P_N, \tag{17}$$

with the generalized projection operator

$$P_N = \sum_{n=0}^{N-1} \left|\phi^l_n\right\rangle \left\langle\phi^l_n\right| \tag{18}$$

Then, using expansion of the solution of the new problem in the basis $\{\phi^l_n\}$, one can find that the tangent of approximated phase shift is given by the following formula, very similar to the relativistic case:

$$\tan\tilde{\delta}_{N,l} = -\frac{s^l_{N-1}(\tilde{k}) + g_{N-1,N-1}(E) J_{N,N-1}(\tilde{k}) s^l_N(\tilde{k})}{c^l_{N-1}(\tilde{k}) + g_{N-1,N-1}(E) J_{N,N-1}(\tilde{k}) c^l_N(\tilde{k})} \tag{19}$$

where s_n^l and c_n^l are coefficients of sine-like and cosine-like solutions of the following equation

$$\left(H_0-\frac{\tilde{k}^2}{2}\right)\sum_{n=0}^{\infty}u_n^l\phi_n^l(\lambda r)=\Omega_u\bar{\phi}_n^l(\lambda r);\quad u=s,c;\quad \Omega_s=0;\quad \Omega_c=-\frac{\tilde{k}}{2s_0^l} \tag{20}$$

Here, $\tilde{k}\equiv\sqrt{\frac{2ME}{\hbar^2}}$ is the wave number related to energy E and mass M of the projectile.

We can choose the same basis functions $\{\phi_n^l\}$ as in the relativistic case. They, and the expansion coefficients s_n^l and c_n^l, are collected in Table 1.

$J_{N,N-1}$ is an element of the following matrix

$$J_{mn}\equiv\left\langle\phi_m^l\right|H_0-\frac{\tilde{k}^2}{2}\left|\phi_n^l\right\rangle\equiv\left\langle\phi_m^l\right|-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2}+\frac{l(l+1)}{2r^2}-\frac{\tilde{k}^2}{2}\left|\phi_n^l\right\rangle. \tag{21}$$

In the above formulas, as in the relativistic case, N is the quantity of base functions ϕ_n^l used to truncate the scattering potential. $g_{N-1,N-1}(E)$ is a matrix element of the inverse of the truncated operator $P_N^\dagger\left(H_0+V^N-\frac{\tilde{k}^2}{2}\right)P_N$, restricted to the N-dimensional space, where it does not vanish. Again, this matrix can be viewed as a matrix approximating the Green function.

2.3. Relativistic cross sections

Once we have calculated the (relativistic) phase shifts for different angular momenta l, we are ready to employ the partial-wave analysis, and calculate the so called direct

$$f(\vartheta)=\frac{1}{2ik}\sum_l\left\{(l+1)\left[\exp\left(2i\delta_{N,l}^+\right)-1\right]+l\left[\exp\left(2i\delta_{N,l}^-\right)-1\right]\right\}P_l(\cos\vartheta) \tag{22}$$

and spin-flip

$$g(\vartheta)=\frac{1}{2ik}\sum_l\left[\exp\left(2i\delta_{N,l}^-\right)-\exp\left(2i\delta_{N,l}^+\right)\right]P_l^1(\cos\vartheta) \tag{23}$$

scattering amplitudes. Then, the differential σ_d and the spin polarization σ_{sp} cross sections may be written as

$$\sigma_d(\vartheta)=|f(\vartheta)|^2+|g(\vartheta)|^2, \tag{24}$$

$$\sigma_{sp}(\vartheta)=\frac{i\left(f(\vartheta)g^*(\vartheta)-f^*(\vartheta)g(\vartheta)\right)}{\sigma_d(\vartheta)} \tag{25}$$

The momentum transfer σ_{mt} and the total σ_t cross sections are defined as:

$$\begin{aligned}\sigma_{mt}(E)=\frac{4\pi}{k^2}\sum_l\Bigg[&\frac{(l+1)(l+2)}{2l+3}\sin^2(\delta_{N,l}^+-\delta_{N,l+1}^+)\frac{l(l+1)}{2l+1}\sin^2(\delta_{N,l}^--\delta_{N,l+1}^-)\\&+\frac{(l+1)(2l+1)}{2l+3}\sin^2(\delta_{N,l}^+-\delta_{N,l+1}^+)\Bigg]\end{aligned} \tag{26}$$

$$\sigma_t(E)=\frac{4\pi}{k^2}\sum_l\left[(l+1)\sin^2\delta_{N,l}^++l\sin^2\delta_{N,l}^-\right] \tag{27}$$

In the above equations, $\delta_{N,l}^-$ corresponds to the solution for $\kappa=l$, and $\delta_{N,l}^+$ for $\kappa=-l-1$.

Non-relativistic formulas for the cross sections can be found elsewhere, *e.g.* in [8].

3. Numerical computations with JMATRIX programs

3.1. A general view

We expect that for $N \to \infty$, the approximate relativistic phase shift $\delta_{N,l}$ (16), or $\tilde{\delta}_{N,l}$ in the non-relativistic case (19) converges to the exact value δ (or $\tilde{\delta}$). Hence, increasing the basis size N, we expect to obtain increasingly accurate results. Having the numerical phase shifts calculated for the selected scattering problem, we are able to use the partial-wave analysis to obtain differential, total, momentum transfer and spin polarization cross sections.

The first program from the JMATRIX package, JMATRIX-CONV, is designed to study the convergence, by systematic calculations of phase shifts for the progressively increased basis size N (*e.g.* $N_{start} = 1 \leq N \leq N_{end} = 1000$), and the chosen angular momentum (*i.e.* $l = 0$, $\kappa = -l-1 = -1$). Optionally, after the basis size $N = N_{end}$ is achieved (which is the most time-consuming part of calculations), phase shifts are calculated for a series of projectile energies.

The second program, JMATRIX-CS, is able to calculate the cross sections for a fixed basis size N, series of angular momenta (*e.g.* $l = 0,1,2$ and equivalent numbers κ), and range of energies. The basis size should be chosen to be large enough to keep the accuracy. A sufficient value of N may be chosen by tracing the convergence using the JMATRIX-CONV program.

In fact, both programs are rather similar to each other – they use similar numerical methods and techniques described below, and share most of the code, so they could be merged into one general tool, nonetheless, we have decided to keep them separate. The main reason for that is the fact that the code has been optimized in many ways for performance, and these optimizations are different for both codes as they are designed for other purposes.

To find the phase shifts, the programs compute all the required mathematical functions (such as the Gegenbauer and Laguerre polynomials, the hypergeometric functions, the Bessel, spherical Bessel, Neumann and spherical Neumann functions and their derivatives, the gamma function and more) to evaluate the basis functions ϕ_n^l and coefficients s_n^l and c_n^l (see Table 1). Then, the programs truncate the scattering potential in the selected basis by numerical integration, and form the matrix approximating the Green function. This matrix is constructed as a sum of the matrix of elements of the truncated potential and elements of the J-matrix. Then, programs inverse this matrix by diagonalization (using some orthogonal matrix; in the present code this orthogonal matrix is chosen to be a matrix of eigenvectors of the Green matrix), and finally, compute an approximate phase shift for a given number N.

Also, for testing purposes, an additional procedure has been written, to calculate phase shifts using an analytical formula for potentials with a shape of a square-well potential. Hence, in the case of the square-well potential, the

numerical results obtained in J-matrix calculations can be directly compared to results obtained using analytical formulas.

Most of the mathematical procedures and functions used in the program have been written by the authors using the useful formulas and relations included in [9] and [10]. Some procedures (for Bessel, Neumann and Gamma[2] functions, for numerical integration and for searching eigenvalues and eigenvectors of real, symmetric matrix) have been taken from public Fortran 77 libraries.

In the relativistic case, most of the formulas are written as function of the total energy of the projectile, E. For user convenience and for unification with the non-relativistic case, we used a rescaled kinetic energy E in our code, instead of the total energy E. Due to this rescaling, the implemented formulas are slightly different than written in section 2. In the non-relativistic case such rescaling is not necessary, because in this case all formulas are written as function of the kinetic energy from the beginning.

Three basis sets, Laguerre, Gaussian and oscillator, are implemented, but in the current version of the codes the latter is available in the non-relativistic calculations only.

Coefficients s_n^l and c_n^l for $N=1$ and $N=2$ were calculated using explicit formulas (see Table 1), but for $N>2$, using three-term recursion relation (see [2]) to avoid numerical instabilities.

In the J-matrix method there are many integrals to be calculated, especially in the relativistic case. As they are used to truncate the scattering potential (17), their quantity is $N\times N$ in the non-relativistic case, and $2N\times 2N$ in the relativistic one. The calculation of these integrals is a major contribution to the computational time. To minimize their quantity (and therefore the overall time of computations), we apply several techniques, such as utilization of the J-matrix symmetry and the scattering potential properties. Moreover, the integrals calculated during the execution of the program are dumped to files, to be used in next runs of the programs, when possible. The current integration method is the Double Exponential (DE) transformation method [11].

Additionally, the JMATRIX-CS code implements formulas (22)–(27) (and their non-relativistic analogues) to obtain cross sections.

3.2. Program structure and compilation

After unpacking the provided gzipped tar file, the following directory structure will be created:

```
jmatrix
|
|--bin
| |--jmatrix-conv
| |--jmatrix-cs
```

2. Although gamma functions and its logarithm are included in many Fortran 95 compilers as intrinsic functions, we decided to use an external library to retain the greatest compatibility.

```
|
|--examples
| |--jmatrix-conv
| |--jmatrix-cs
|
|--src
|--jmatrix-conv
|--jmatrix-cs
|--lib
```

The subdirectories `jmatrix/src/jmatrix-conv` and `jmatrix/src/jmatrix-cs` contain source files and `Makefiles` for the JMATRIX-CONV and JMATRIX-CS programs, respectively, while common libraries are stored in the `jmatrix/src/lib`. By analogy, the executables created during the compilation process are placed, together with input and output files in the subdirectories `jmatrix/bin/jmatrix-*`, and example input and test run output files are stored in the `jmatrix/examples/jmatrix-*` subdirectories.

For clarity, source codes have been split into several files. Every file contains a code responsible for a specific task, *e.g.* the procedures included in the file `jm_specfun.f95` calculate the required special functions, `jm_v.f95` truncates the scattering potential, `jm_analytic.f95` calculates the analytical value of the phase shift in case of the square-well potential, *etc.* Although most of the filenames in both the codes are the same, they usually differ in implementation, as they have been designed for separate tasks.

The main segments of the programs are placed in the files `jmatrix-conv.f95` and `jmatrix-cs.f95`. There are two special modules: `jm_constants.f95` and `jm_globals.f95`. The former defines some constants and user types used in program, the latter defines global variables.

This program is written in ANSI/ISO Fortran 95, hence, it is expected to be compiled with any Fortran 95 or Fortran 2003/2008 compilers without any problem. Additionally, the program uses some numerical libraries written in Fortran 77, they also can be successfully compiled with the use of the same Fortran 95 versions or a newer compiler.

To make the compilation as easy as possible, `Makefiles` are supplied, separately for both codes. They are placed together with the sources. Before compilation, the user should edit `Makefile`, specifying the compiler type/name, the operating system, the target platform, the parallelization and optimization options and the linking method. `Makefiles` are well-commented, so the user can easily adapt them to a specific environment.

After editing `Makefiles`, the compilation can be started as usual, by typing the command `make` inside the selected directory `jmatrix/source/jmatrix-*`. In case of successful compilation, the executable file `jmatrix-conv` or `jmatrix-cs` (with suffix `.exe` when compiling under Windows) will be created and moved[3] to the directory `jmatrix/bin/jmatrix-*`.

3. The GNU Make tool for Windows has a bug, preventing calling the shell commands from within Makefile. As a result, the compiled executable cannot be automatically moved. Please

To remove all files created during the compilation process, use the command `make clean`. Please note that the executables, previously moved to the `bin/*` subdirectories, will not be deleted.

See the provided `README` file for additional information about the compilation process.

3.3. Input and output files

In both programs, all calculations are controlled through the parameters read from the external text file. The name of the input file is arbitrary, since the program asks for it at the start of execution. The default names are `jmatrix-conv.inp` and `jmatrix-cs.inp`. The input file should be composed of a set of lines in the following form:

```
; comment keyword = value ; comment
```

Blank lines, lines beginning with „`;`" and lines without proper keywords are ignored. Both keywords and values are case sensitive, and their order in the file is arbitrary. Not all keywords are required in all cases, this will be discussed later. Please note that there *must* be at least one space around the „`=`" sign and before the „`;`" sign (except the case when whole line is a comment).

Full example input files and output files created during the execution of the codes are included in the provided archive in the subdirectories `jmatrix/example/jmatrix-*`.

3.3.1. JMATRIX-CONV – example input file and description of output files

First, we describe the input file for the `JMATRIX-CONV` program. In Table 2 we present an example input file, containing *all* keywords recognized by the program. The meanings of individual parameters are discussed later, but they can be easily presumed from their names and included comments.

First, the orbital angular momentum quantum number l and the quantum number κ, which describe the projectile, have to be specified, under the condition that $\kappa = l$ or $\kappa = -l-1$. To specify these numbers, the keywords `l` and `kappa`, respectively, should be used. If proper dependency between κ and l is not satisfied, the program stops with an error message. This restriction matters only in relativistic calculations as κ is not used in the non-relativistic scheme – the keyword `kappa` is ignored in that case.

Next, the user should provide the (positive) kinetic energy of the projectile, E (in atomic units), by specifying the parameter `e`. Additionally, the program can calculate phase shifts for projectile energies in the range $E \in [\texttt{estart}, \texttt{eend}]$ ($\texttt{estart} \leq \texttt{eend}$) with the step size `estep` in a single run of the program. These calculations are performed only if flag `energies` is set to `.true.`. This energy range may be completely different than the energy specified before. When no `estep` is specified, the program takes $\texttt{estep} = (\texttt{eend} - \texttt{estart})/10$.

move the created executable manually to the proper `jmatrix/bin/jmatrix-*` subdirectory, or use the provided `compile-win.bat` script, which bypasses the bug.

Table 2. Example input file for the JMATRIX-CONV program; descriptions and comments are shortened, to save space

```
; PROPERTIES OF THE PROJECTILE
l = 0 ; Orbital angular momentum quantum number
kappa = -1 ; Relativistic quantum number
e = 0.4 ; Energy of the projectile
energies = .true. ; Calculate phase shifts for many energies at once?
estart = 0.2 ; If above flag is set to true, additional calculations
eend = 1.0 ; are performed for a~set of energies within the range
estep = 0.2 ; [estart, eend] (in a. u.), with step size estep.

; TYPE OF CALCULATIONS
lambda = 1.0 ; Scaling parameter.
basis = laguerre ; Basis set (gauss, laguerre, oscillator).
scheme = relativistic ; Scheme (relativistic, non-relativistic)
v_light = finite ; Velocity of light (finite, infinite)
                 ; in relativistic calculations.
nstart = 1 ; Initial value of basis size N.
nend = 800 ; Final value of basis size N.
napprox = 0 ; Basis size, from what the approximations start.

; SCATTERING POTENTIAL
pot_type = coulomb ; Potential type (well, coulomb, yukawa, user).
r0 = 1.0 ; Truncating parameter, so that v(r > r0) = 0.

; PARAMETERS OF POTENTIAL SQUARE-WELL
v0 = -1.0 ; Depth.
a = 0.8 ; Left bound.
b = 1.0 ; Right bound.

; PARAMETERS OF COULOMB-LIKE POTENTIAL V(r) = - z / (r^alpha)
z = 30.0 ; The charge parameter.
alpha = 1.0 ; The power parameter.

; PARAMETERS OF YUKAWA POTENTIAL V(r) = - g^2 * Exp(-m*r) / r
g = 1.0 ; Coupling constant.
m = 0.5 ; Mass parameter.

; OTHER PARAMETERS
screen = .true. ; Display (or not) results and other information on screen.
shift = .false. ; Shift (or not) calculated phase shifts to range [0, pi].
```

One can specify the scaling parameter λ (keyword `lambda`). If omitted, the default value $\lambda = 1.0$ will be taken. Changing this parameter may improve the convergence in the oscillator basis. In this case, the quantity $1/\lambda$ is equivalent to the oscillator length. It is recommended to leave this parameter untouched in case of the Laguerre and Gaussian bases as these bases are biorthonormal, and its influence has not been investigated in details yet. Moreover, due to numerical reasons (see the formula for c_n^l in Table 1), it is not allowed to put the value 0.5 for `lambda` when using the Laguerre basis.

The J-matrix (6) and the truncated potential (13) elements can be calculated in three different basis sets, Laguerre, Gauss or oscillator. This is determined through the keyword `basis`. It can assume three values, `laguerre`, `gauss`

or `oscillator`. Also, the `scheme` of computations (`relativistic` or `non-relativistic`) should be specified. If no keyword `scheme` is found in the input file, calculations will be performed using the non-relativistic scheme.

We introduced an additional keyword, `v_light` to verify that the relativistic results converge to the non-relativistic limit as the speed of light approaches infinity. The standard value is `finite`, it is responsible for the constant and finite speed of light (137.036 in atomic units). By specifying `v_light = infinite`, one can get a non-relativistic limit in relativistic computations and compare it with pure non-relativistic calculations. This setting should not be used in real calculations, it has been added only for testing purposes. The keyword `v_light` is ignored in the non-relativistic scheme of calculations.

The initial (`nstart`) and final (`nend`) values of the basis size N should be specified, so that the calculations should be performed for $\texttt{nstart} \leq N \leq \texttt{nend}$, with step 1. It is useful to study the convergence of the phase shifts, while increasing the basis size. If `nstart` is omitted, it is taken as 1. When $\texttt{nstart} = \texttt{nend}$, calculations are performed for only one value of $N = \texttt{nstart} = \texttt{nend}$. The latter case does not allow studying the convergence, but it may be useful in case when one needs only to calculate the phase shifts for different energies with the use of previously saved elements of the truncated potential. These elements are automatically saved to a file named (`saved.vn.<nend>`) when `nend` is achieved, and then read in further runs of the program, when possible. As a consequence, it is possible to continue already finished calculations, by specifying new `nstart` as `nend`, taken from the finished calculations.

The parameter `napprox` is used to speed-up the non-relativistic calculations. When its value is positive, we use the approximated formula for elements of the truncating potential for $\texttt{napprox} < N \leq \texttt{nend}$. This approximation has been proposed in [12]. One should put 0 or a negative number to disable approximations.

The keyword `pot_type` is responsible for the kind of scattering potential used in computations. There are three types of potentials predefined in the program: square-well (value: `well`), the truncated Coulomb potential (value: `coulomb`) and the Yukawa potential (value: `yukawa`). The specific parameters for the above potentials should also be specified in the input file.

If square-well is selected, the depth `v0`, left and right bounds (`a` and `b`, respectively) are required (all of them in atomic units), according to the following definition:

$$V(r) = \begin{cases} 0 & \text{for} \quad r \in (0,a) \\ V_0 & \text{for} \quad r \in [a,b) \\ 0 & \text{for} \quad r \in [b,\infty) \end{cases} \tag{28}$$

The Coulomb potential has the form $V(r) = -z/r^{\alpha}$, so the parameters `z` and `alpha` should be specified. The Yukawa potential has the form $V(r) = -g^2 \exp(-mr)/r$, so the parameters `g` and `m` should be specified. In the Coulomb and Yukawa potentials, the parameter `r0` (in atomic units) specifies at which r the potential will be truncated, so that $V(r > \texttt{r0}) = 0$.

Moreover, the program is able to use a user-defined scattering potential given in any analytical form, by using the value `other` for the keyword `pot_type` in the input file. The requested formula describing the potential should be specified in the file `user_potential`, then the program has to be recompiled.

The last two parameters, `screen` and `shift` are of a logical-type and they are responsible for displaying (default) or not the results on screen and for shifting (or not, which is default) the calculated phase shifts to the range $[0,\pi]$.

During its execution, the program creates two or more output files with the results of calculations.

- The main file is named `phases.<suffix>.out`, where `suffix` is optional. The program asks for this suffix at the beginning of its execution, just after collecting the name of the input file. In that file, the calculated phase shifts as function of the basis size are saved, for the specified energy and quantum numbers.
- Second output file, `phases-rms.<suffix>.out`, is created only when the potential square-well has been used and there are at least 200 phase shifts calculated (`nend`$-$`nstart`≥ 200). It contains the root-mean-square error of the calculated phase shifts, with the use of the analytical phase shifts.
- Third output file, `phases-avg.<suffix>.out`, is created only when there are at least 50 phase shifts calculated (`nend`$-$`nstart`≥ 50), and contains averaged phase shifts. Averaging is done over every 50 calculated phase shifts.
- Next output file, `phases-en.<suffix>.out` is created when the flag `energies = .true.` and it contains calculated phase shifts as function of energy of the projectile (for the final value of N, specified by the `nend` keyword).

Additionally, the file `saved.vn.<nend>` is created. It contains elements of the truncated potential. This file is utilized to save the computational time, in separate runs of the program. The program will automatically use this file, when information stored in that file corresponds to the actual set of parameters, *i.e.* calculations are performed in the same basis, scheme, with the same quantum numbers describing the projectile and the scaling parameter. When this information does not conform to the current calculations, the file `saved.vn.<nend>` is deleted and created again for a new set of parameters. For example, when the current calculations were performed for `nstart`$=1$ and `nend`$=500$, it was possible to start new calculations with `nstart`$=500$ and any `nend`$\geq$`nstart`. It saves a lot of time.

At the beginning of each output file, a header containing the parameters used to calculations is written. It allows output files to be easily distinguished for several scattering systems.

3.3.2. JMATRIX-CS – example input file and description of output files

Table 3 shows an example of the input file containing *all* keywords recognized by the program.

Now, we emphasize only the differences between the input file for the `JMATRIX-CS` code, and the previously described input file for `JMATRIX-CONV`.

Table 3. Example input file for the JMATRIX-CS program; descriptions and comments are shortened, to save space

```
; PROPERTIES OF THE PROJECTILE AND PARTIAL WAVE EXPANSION
estart = 0.2 ; Calculations are performed for a~set of energies within range
eend = 1.0 ; [estart, eend] (in atomic units),
estep = 0.2 ; with step size estep.
lmax = 1 ; Upper limit of angular momentum quantum numbers
         ; for partial waves expansion.

; TYPE OF CALCULATIONS
lambda = 1.0 ; Scaling parameter.
basis = laguerre ; Basis set (laguerre, gauss, oscillator).
scheme = relativistic ; Scheme (relativistic, non-relativistic).
v_light = finite ; Velocity of light (finite, infinite)
                 ; in relativistic calculations.
ntrunc = 800 ; Basis size to truncate scattering potential.
napprox = 0 ; Basis size, from what the approximations start.

; SCATTERING POTENTIAL
pot_type = coulomb ; Potential type (well, coulomb, yukawa, user).
r0 = 1.0 ; Truncating parameter, so that v(r > r0) = 0.

; PARAMETERS OF POTENTIAL SQUARE-WELL
v0 = -1.0 ; Depth.
a = 0.8 ; Left bound.
b = 1.0 ; Right bound.

; PARAMETERS OF COULOMB-LIKE POTENTIAL V(r) = - z / (r^alpha)
z = 30.0 ; The charge parameter.
alpha = 1.0 ; The power parameter.

; PARAMETERS OF YUKAWA POTENTIAL V(r) = - g^2 * Exp(-m*r) / r
g = 1.0 ; Coupling constant.
m = 0.5 ; Mass parameter.

; OTHER PARAMETERS
screen = .true. ; Display (or not) results and other information on screen.
shift = .false. ; Shift (or not) calculated phase shifts to range [0, pi].
```

Here, we do not have a separate parameter for specifying the projectile energy. The only parameters for the projectile energy are now `estart`, `eend` and `estep`, of the same meaning as before. Also, there is no need to set the flag `energies`.

The parameters to specify the orbital angular momentum quantum number l and the quantum number κ are removed. Instead, the user should specify the maximum orbital angular momentum quantum number (keyword `lmax`), taken into account in the partial-waves analysis. For example, when `lmax` is set to 3, the relativistic phase shifts will be calculated for the following pairs of (l,κ): $(0,-1)$, $(1,-2)$, $(1,1)$, $(2,-3)$, $(2,2)$, $(3,-4)$, $(3,3)$.

As the `JMATRIX-CS` code has been designed for calculations in a fixed basis size (of an optimal value found *e.g.* by using the `JMATRIX-CONV` code), instead a set of parameters `nstart`, `nend`, we have only one parameter specifying the basis size: `ntrunc`.

As previously, the program asks for this suffix at the beginning of its execution, after collecting the name of the input file. The following output files are now created:

- `phases.<suffix>.out` with numerical phase shifts for a given basis size, a set of quantum numbers and a range of projectile energies;
- `dcs.<suffix>.out` with differential cross sections for a range of energies;
- `tcs.<suffix>.out` with total cross sections;
- `mtcs.<suffix>.out` with momentum transfer cross sections;
- `spcs.<suffix>.out` with spin polarization cross sections (only in a relativistic case).

As in the previous code, elements of the truncated potential are dumped to files, now named `saved.vn.<ntrunc>.<l>.<kappa>`. It allows repeating calculations *e.g.* for a different set of energies without time-consuming integrations.

3.4. Results and discussion

The JMATRIX codes were thoroughly tested with many combinations of input parameters. In each case numerical stability was achieved.

Here, we present the results obtained with the input parameters as presented in Tables 2 and 3. All calculations were performed for the truncated Coulomb scattering potential, in the Laguerre basis.

In Figure 1 we can observe the convergence of phase shifts versus the basis size. As we can see, the convergence for this case is rather slow but systematic. Also, the result is in good agreement with the result of Krośnicki [13].

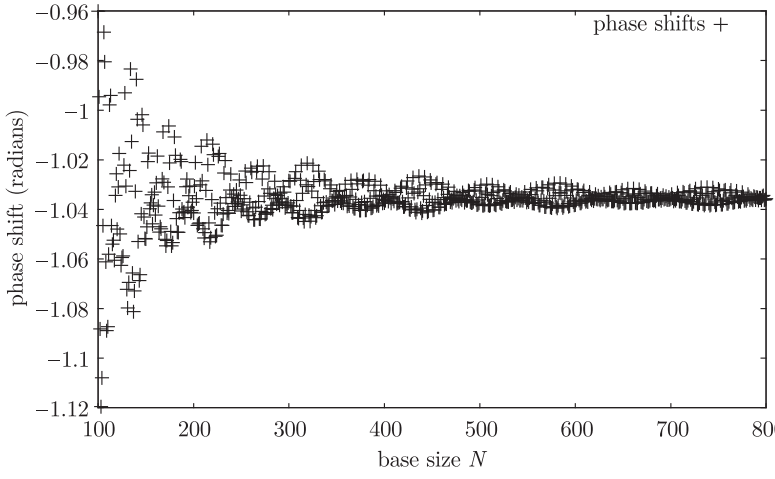

Figure 1. Convergence of relativistic phase shift versus the number of Laguerre basis function N, used to truncate the scattering potential; $l=0$, $\kappa=-1$, $E=0.4$ a.u.
result from [13]: -1.035866

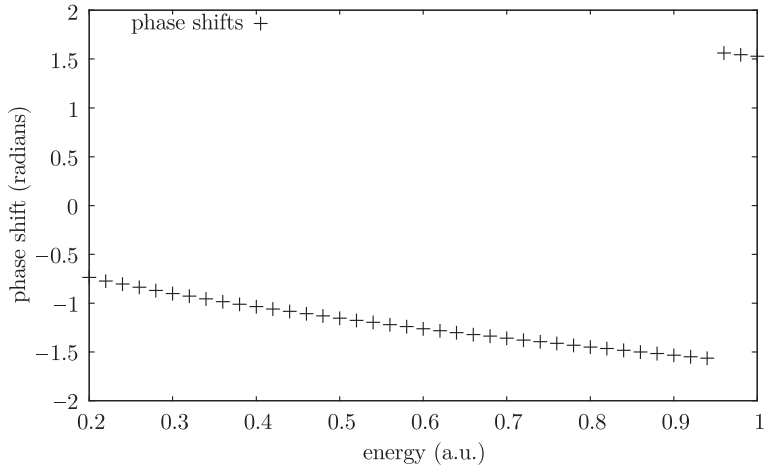

Figure 2. Numerical phase shift versus the projectile energy; $l = 0$, $\kappa = -1$, $E = 0.4$ a.u.

Phase shifts for a set of projectile energies are presented in Figure 2. The results obtained by using the input parameters from Table 2, were completed by calculations for more intermediate energies.

In Figures 3 and 4, differential and spin polarization cross sections in scattering of electrons from the truncated Coulomb potential are plotted for different electron energies.

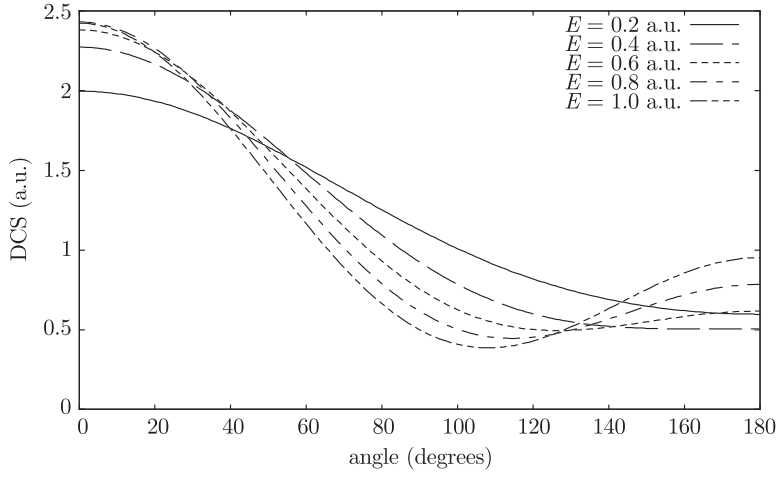

Figure 3. Differential cross sections in elastic scattering of electrons from truncated Coulomb potential

Total and momentum transfer cross sections are presented in Figure 5. The results obtained by using the input parameters from Table 2 were completed by calculations for more intermediate energies.

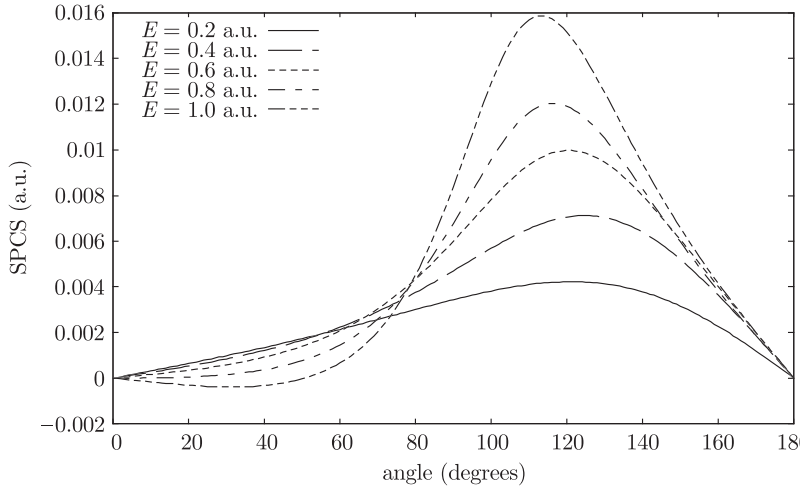

Figure 4. Spin polarization cross sections in elastic scattering of electrons from truncated Coulomb potential

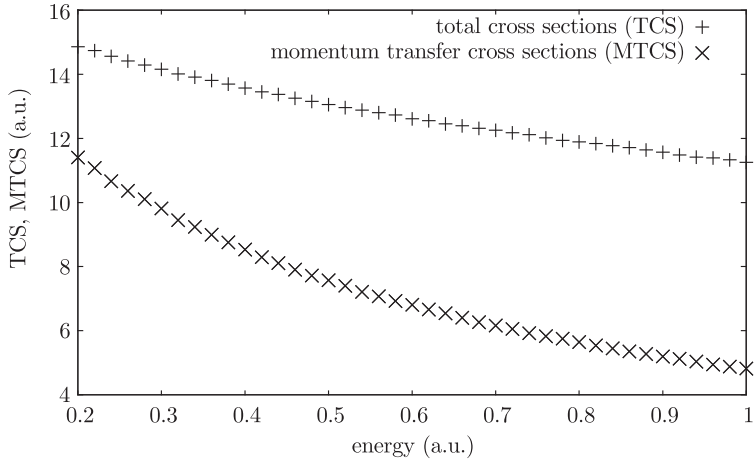

Figure 5. Total and momentum transfer cross sections in elastic scattering of electrons from truncated Coulomb potential

4. Conclusions

In the last few years the interest in the J-matrix method has significantly increased, mainly due to relativistic extension of the method and its application to the photoionization ([14, 15]). Also, the multichannel extension has been proposed and tested ([16]). Comparing to other methods, *i.e.* multicofiguration Dirac-Fock ([17, 18]), the method allows calculations for several incident energies with a relatively small computational effort.

Here, we propose a set of programs for elastic scattering phase shifts calculations using the J-matrix method, both relativistic as well as non-relativistic versions. The calculated phase shifts are then used to calculate differential, total, spin polarization and momentum transfer cross sections.

The presented package allows applying any scattering potential vanishing faster than the Coulomb potential and given in an analytical form. An example of scattering on the truncated Coulomb potential has been presented.

There are several ways for our further investigations of the J-matrix method and developing the JMATRIX package. The first method will consist in adding the atomic targets in elastic scattering. Then, inelastic scattering will be introduced. The second way will consist in adaptation of the JMATRIX package to use in multiphoton single and double ionization of atoms.

Appendix A: JMATRIX-CONV – test run output

The following is a partial listing of the output file `phases.out` with phase shifts[4] versus the basis size, produced by running the program JMATRIX-CONV with the input data file displayed in Table 2. The result is plotted in Figure 1.

```
; OUTPUT FROM THE JMATRIX-CONV PROGRAM
; date: 06.03.2017 time: 15.54
;
; energy   =        0.4000000000000000
; l        =        0
; kappa    =       -1
; lambda   =        1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    = F
; nstart   = 1
; nend     = 800
; napprox  = 0
; pot_type = coulomb
; r0       =        1.0000000000000000
; z        =       30.0000000000000000
; alpha    =        1.0000000000000000
;
; NUMERICAL RESULTS:  N  tan(delta)  delta
    1       1.4436859860002820       0.9650058354002918
    2      -7.2935796275664940      -1.4345390030771520
    3      -0.7577541953878013      -0.6484453508657673
    4       0.3325885835580812       0.3210801299097921
    5       3.0719824293835960       1.2560917404464580
    6      -2.1400000250798780      -1.1336576625694830
    7      -0.6960516717147044      -0.6080711614140711
    8      -0.3217172058928592      -0.3112598648694454
    9      -0.3681945644029319      -0.3527909530536955
   10      -1.6265146513012670      -1.0195571047193800
(...)
```

4. Please note that last digits in the presented phase shifts may be slightly different when using different compilers.

```
  100      -1.4284853174042920      -0.9600420428225145
  101      -1.5388742059853840      -0.9945436504268680
  102      -1.9087989519037480      -1.0882201651185770
  103      -2.0637600363004700      -1.1195829451367280
  104      -2.0039861673690500      -1.1079446819609800
  105      -1.7295557811174060      -1.0465731200430290
  106      -1.4546108779256980      -0.9685299779735516
  107      -1.4925573967605910      -0.9804958020433222
  108      -1.7894447585046440      -1.0611971784602960
  109      -1.9121056186830640      -1.0889313028665690
  110      -1.9047956044283380      -1.0873566066322920
(...)
(...)
  300      -1.6593440729878360      -1.0284322131643040
  301      -1.6601840782995450      -1.0286559272320880
  302      -1.6981877146925300      -1.0386060071849260
  303      -1.7090324456313380      -1.0413850913174440
  304      -1.7087974808680580      -1.0413251572948720
  305      -1.6985835208550830      -1.0387079005479220
  306      -1.6560287856975640      -1.0275476465101900
  307      -1.6458461447140100      -1.0248144756195580
  308      -1.6898256681119600      -1.0364450746449170
  309      -1.7164113678855730      -1.0432610532078590
  310      -1.7180313639971640      -1.0436712983703760
(...)
  500      -1.6751488683172080      -1.0326137405885180
  501      -1.6661096819263750      -1.0302293531498250
  502      -1.6849510898408280      -1.0351780538483590
  503      -1.6976027977302150      -1.0384553659737790
  504      -1.6976334345608100      -1.0384632581846120
  505      -1.6960592917715350      -1.0380574767307470
  506      -1.6782823834603120      -1.0334358890826290
  507      -1.6645613752004400      -1.0298190254584810
  508      -1.6807206973302300      -1.0340740701128630
  509      -1.6968863496183180      -1.0382707448067310
  510      -1.6978305766513450      -1.0385140379794300
(...)
  790      -1.6843694372918920      -1.0350265060623720
  791      -1.6848829126012750      -1.0351602944810790
  792      -1.6895199059344910      -1.0363657587838310
  793      -1.6857696019189820      -1.0353911836483270
  794      -1.6854758830781660      -1.0353147207309590
  795      -1.6889487947394440      -1.0362175534322880
  796      -1.6834292264780580      -1.0347813733047800
  797      -1.6820119641393190      -1.0344114804475810
  798      -1.6889014613241090      -1.0362052669151110
  799      -1.6882777772045880      -1.0360433271862530
  800      -1.6869463881640860      -1.0356973351931200
```

The following is a partial listing of the output file `phases-en.out` with phase shifts versus the projectile energy, produced by running the program JMATRIX-CONV with the input data file displayed in Table 2. The results completed by calculations for more intermediate energies, are plotted in Figure 2

```
; OUTPUT FROM THE JMATRIX-CONV PROGRAM
; date: 06.03.2017 time: 17.37
;
; estart   = 0.2000000000000000
```

```
; eend     = 1.0000000000000000
; estep    = 0.2000000000000000
; l        = 0
; kappa    = -1
; lambda   = 1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    = F
; ntrunc   = 800
; pot_type = coulomb
; r0       = 1.0000000000000000
; z        = 30.0000000000000000
; alpha    = 1.0000000000000000
;
; NUMERICAL RESULTS: energy tan(delta) delta
  0.200000      -0.9048125148028265      -0.7354675961868376
  0.400000      -1.6869463881640860      -1.0356973351931200
  0.600000      -3.1274310464893210      -1.2613190386478510
  0.800000      -8.1805304614505160      -1.4491583474052740
  1.000000      24.1538661817417700       1.5294187177135440
```

The following is a partial listing of the output file `phases-avg.out` with phase shifts averaged over every 50 results, versus the projectile energy, produced by running the program JMATRIX-CONV with the input data file displayed in Table 2.

```
; OUTPUT FROM THE JMATRIX-CONV PROGRAM
; date: 06.03.2017 time: 17.37
;
; energy   = 0.4000000000000000
; l        = 0
; kappa    = -1
; lambda   = 1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    = F
; nstart   = 1
; nend     = 800
; napprox  = 0
; pot_type = coulomb
; r0       = 1.0000000000000000
; z        = 30.0000000000000000
; alpha    = 1.0000000000000000
;
; AVERAGED RESULTS:  N  averaged_delta
   501      -0.626294566087534600
  1001      -0.720398615846079800
  1501      -1.042201442840791000
  2001      -1.036108062525964000
  2501      -1.035508879706652000
  3001      -1.035482248949317000
  3501      -1.035448656548142000
  4001      -1.035452544304233000
  4501      -1.035355437368863000
  5001      -1.035526401969362000
  5501      -1.035451537272408000
```

```
6001      -1.035540244238388000
6501      -1.035350284693187000
7001      -1.035565732996265000
7501      -1.035505095437630000
8001      -1.035509634873714000
```

Appendix B: JMATRIX-CS – test run output

The following is a listing of the output file `phases.out` with phase shifts versus quantum numbers and the projectile energy, produced by running the program JMATRIX-CS with the input data file displayed in Table 3.

```
; OUTPUT FROM THE JMATRIX-CS PROGRAM
; date: 06.03.2017 time: 22.27
;
; estart   =       0.2000000000000000
; eend     =       1.0000000000000000
; estep    =       0.2000000000000000
; lmax     =       1
; lambda   =       1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    = F
; ntrunc   = 800
; napprox  = 0
; pot_type = coulomb
; r0       =        1.0000000000000000
; z        =       30.0000000000000000
; alpha    =        1.0000000000000000
;
; PHASE SHIFTS: l kappa delta
;
; e =      0.20000
;
  0      -1       -0.7354675961655643
  1      -2       -0.0888352741605350
  1       1       -0.0866645765767718
;
; e =      0.40000
;
  0      -1       -1.0356973351309860
  1      -2       -0.2070842885111708
  1       1       -0.2036286954886156
;
; e =      0.60000
;
  0      -1       -1.2613190387931790
  1      -2       -0.3236089801259628
  1       1       -0.3186842589748248
;
; e = 0.80000
;
  0      -1       -1.4491583472968800
  1      -2       -0.4375655918415151
  1       1       -0.4314227779877501
;
```

```
; e =      1.00000
;
  0      -1        1.5294187177582880
  1      -2       -0.5454001318105261
  1       1       -0.5370995776847606
```

The following is a partial listing of the output file `dcs.out` – differential cross sections, produced by running the program JMATRIX-CS with the input data file displayed in Table 3. The result is plotted in Figure 3.

```
; OUTPUT FROM THE JMATRIX-CS PROGRAM
; date: 06.03.2017 time: 22.27
;
; estart   =       0.2000000000000000
; eend     =       1.0000000000000000
; estep    =       0.2000000000000000
; lmax     =       1
; lambda   =       1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    =  F
; ntrunc   = 800
; napprox  = 0
; pot_type = coulomb
; r0       =       1.0000000000000000
; z        =      30.0000000000000000
; alpha    =       1.0000000000000000
;
; DIFFERENTIAL CROSS SECTIONS: theta DCS
;
; e = 0.20000
;
    0       1.997028770204161
    1       1.996869917561556
    2       1.996393455717600
    3       1.995599672836478
    4       1.994489048900678
    5       1.993062255274577
    6       1.991320154094161
    7       1.989263797483546
    8       1.986894426599160
    9       1.984213470502679
   10       1.981222544863948
(...)
  170       0.6019666475794693
  171       0.6009232148378477
  172       0.5999911761660854
  173       0.5991700164475436
  174       0.5984592784870765
  175       0.5978585640120654
  176       0.5973675345427406
  177       0.5969859121306079
  178       0.5967134799639825
  179       0.5965500828398124
  180       0.5964956275011480
;
; e = 0.40000
;
```

```
    0       2.274683004903255
    1       2.274406597892534
    2       2.273577590524305
    3       2.272196623567552
    4       2.270264764228106
    5       2.267783505035843
    6       2.264754762288580
    7       2.261180874054479
    8       2.257064597735305
    9       2.252409107193364
   10       2.247217989445482
(...)
  170       0.5050549135101174
  171       0.5051501602023247
  172       0.5052420116899238
  173       0.5053282710847509
  174       0.5054069921066956
  175       0.5054764820157850
  176       0.5055353041609367
  177       0.5055822801421263
  178       0.5056164915832077
  179       0.5056372815131269
  180       0.5056442553537605
  (\etc. for e = 0.6, 0.8 and 1.0)
```

The following is a partial listing of the output file `spcs.out` – spin polarization cross section, produced by running the program JMATRIX-CS with the input data file displayed in Table 3. The result is plotted in Figure 4

```
; OUTPUT FROM THE JMATRIX-CS PROGRAM
; date: 06.03.2017 time: 22.27
;
; estart   =       0.2000000000000000
; eend     =       1.0000000000000000
; estep    =       0.2000000000000000
; lmax     =       1
; lambda   =       1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    = F
; ntrunc   = 800
; napprox  = 0
; pot_type = coulomb
; r0       =       1.0000000000000000
; z        =      30.0000000000000000
; alpha    =       1.0000000000000000
;
; SPIN POLARIZATION CROSS SECTIONS: theta SPCS
;
; e = 0.20000
;
    0       0.000000000000000
    1       3.1619710330447731E-005
    2       6.3246876428162775E-005
    3       9.4888950596280718E-005
    4       1.2655337819537005E-004
    5       1.5824759413637761E-004
    6       1.8997901933053251E-004
```

```
     7       2.2175505708184350E-004
     8       2.5358308940771142E-004
     9       2.8547047327305211E-004
    10       3.1742453672327608E-004
(...)
   170       1.1867857928092216E-003
   171       1.0711838638016904E-003
   172       9.5461680604832806E-004
   173       8.3718827718466345E-004
   174       7.1900319653249499E-004
   175       6.0016758837562240E-004
   176       4.8078842476774328E-004
   177       3.6097346799096087E-004
   178       2.4083111276035376E-004
   179       1.2047022825007478E-004
   180       0.000000000000000
;
; e = 0.40000
;
     0       0.000000000000000
     1       2.5096047654032672E-005
     2       5.0210306429317313E-005
     3       7.5360998942991230E-005
     4       1.0056637079129402E-004
     5       1.2584470200729874E-004
     6       1.5121431847997684E-004
     7       1.7669360332104190E-004
     8       2.0230100816535774E-004
     9       2.2805506438989284E-004
    10       2.5397439423528880E-004
(...)
   170       1.8597794515729695E-003
   171       1.6760652124205102E-003
   172       1.4916187060227623E-003
   173       1.3065308952993420E-003
   174       1.1208894161784198E-003
   175       9.3477895477317430E-004
   176       7.4828164044765698E-004
   177       5.6147745221384960E-004
   178       3.7444463602963617E-004
   179       1.8726013067811361E-004
   180       0.000000000000000
(\etc. for e = 0.6, 0.8 and 1.0)
```

The following is a listing of the output file `tcs.out` – total cross section, produced by running the program JMATRIX-CS with the input data file displayed in Table 3. The result is plotted in Figure 5.

```
; OUTPUT FROM THE JMATRIX-CS PROGRAM
; date: 06.03.2017 time: 22.27
;
; estart   =       0.2000000000000000
; eend     =       1.0000000000000000
; estep    =       0.2000000000000000
; lmax     =       1
; lambda   =       1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
```

```
; shift    = F
; ntrunc   = 800
; napprox  = 0
; pot_type = coulomb
; r0       =       1.0000000000000000
; z        =      30.0000000000000000
; alpha    =       1.0000000000000000
;
; TOTAL CROSS SECTIONS: energy TCS
  0.2000000000000000       14.85988263646333
  0.4000000000000000       13.57214926579226
  0.6000000000000001       12.61534597658613
  0.8000000000000000       11.89499924267921
  1.000000000000000        11.25266236617469
```

The following is a listing of the output file `mtcs.out` – momentum transfer cross section, produced by running the program JMATRIX-CS with the input data file displayed in Table 3. The result is plotted in Figure 5.

```
; OUTPUT FROM THE JMATRIX-CS PROGRAM
; date: 06.03.2017 time: 22.27
;
; estart   =        0.2000000000000000
; eend     =        1.0000000000000000
; estep    =        0.2000000000000000
; lmax     =        1
; lambda   =        1.0000000000000000
; basis    = laguerre
; scheme   = relativistic
; v_light  = finite
; shift    = F
; ntrunc   = 800
; napprox  = 0
; pot_type = coulomb
; r0       =        1.0000000000000000
; z        =       30.0000000000000000
; alpha    =        1.0000000000000000
;
; MOMENTUM TRANSFER CROSS SECTIONS: energy MTCS
  0.2000000000000000       11.40419361702000
  0.4000000000000000        8.531863741223996
  0.6000000000000001        6.806331384893004
  0.8000000000000000        5.643419652637525
  1.000000000000000         4.817551883954241
```


Acknowledgements

The calculations were carried out at the Academic Computer Centre in Gdańsk.

The authors thank T. Sumpter for testing the code. This work was partially supported by Grant No 645/N-COST/2010/0 from the Polish Ministry of Science and Higher Education and the COST action CM0702.

References

[1] Heller E and Yamani H 1974 *Phys. Rev. A* **9** 1201

[2] Heller E and Yamani H 1974 *Phys. Rev. A* **9** 1209

[3] Yamani H and Fishman L 1975 *J. Math. Phys.* **16** 410

[4] Horodecki P 2000 *Phys. Rev. A* **62** 52716
[5] Alhaidari D, Yamani H A, and Abdelmonem M S 2001 *Phys. Rev. A* **63** 62708
[6] Syty P 1999 *TASK Quarterly* **3** (3) 269
[7] Yamani H A, Alhaidari A D and Abdelmonem M S 2001 *Phys. Rev. A* **64** 42703
[8] Sin Fai Lam L T 1980 *Aust. J. Phys.* **33** 261
[9] Bateman H and Erdely A (ed.) 1953 *Higher Transcendental Functions*, McGraw-Hill, New York, **I,II**
[10] Arfken G 1970 *Mathematical Methods For Physicists*, Academic Press, New York
[11] Takahasi H and Mori M 1974 *Publ. RIMS*, Kyoto Univ., **9** 721
[12] Vanroose W, Broeckhove J, and Arickx F 2002 *Phys. Rev. Lett.* **82** 10404
[13] Krośnicki M 1988 *MSc thesis*, Gdańsk University of Technology
[14] Foumouo E, Lagmago Kamta G, Edah G and Piraux B 2006 *Phys. Rev. A* **74** 63409
[15] Foumouo E, Antoine P, Bachau H and Piraux B 2008 *New J. Phys.* **10** 25017
[16] Syty P, Redynk Ł, and Sienkiewicz J E 2013 *Eur. Phys. J. Special Topics* **222** 2323
[17] Syty P, Sienkiewicz J E and Fritzsche S 2003 *Rad. Phys. Chem.* **68** 301
[18] Syty P and Sienkiewicz J E 2005 *J. Phys. B: At., Mol. and Opt. Phys.* **38** 2859