Computer Physics Communications 76 (1993) 80–86
North-Holland

Computer Physics Communications

CONVUS – an efficient package for calculating three-dimensional convolution-type integrals

I.E. Lagaris
Physics Department, University of Ioannina, 45110 Ioannina, Greece

and

D.G. Papageorgiou [1]
Department of Chemistry, University of Ioannina, 45110 Ioannina, Greece

Received 4 December 1992; in revised form 18 December 1992

We present two versions of a program and the underlying algorithm, for the efficient calculation of integrals of the form $Z(r_1, r_2) = \int d^3r_3\, p(r_3)\, X(r_1, r_3)\, Y(r_2, r_3)$. The two versions differ in the amount of required memory. Such integrals appear in many calculations in physics and chemistry, specially when studying many-body problems of finite systems.

PROGRAM SUMMARY

Title of program: CONVUS – LOW MEMORY

Catalogue number: ACNI

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Licensing provisions: none

Computer for which the program is designed and others on which it has been tested:
Computers: (i) Control Data CD-4680, (ii) CRAY XMP, (iii) IBM RISC System/6000, (iv) IBM 3090, (v) VAX 8350, (vi) Macintosh II ci; *Installation*: University of Ioannina, Ioannina, Greece

Operating system: (i) EP/IX 1.4.3, (ii) COS, (iii) A/IX, (iv) VM/CMS, (v) VMS, (vi) System 7/MPW

Programming language used: ANSI Fortran 77 (ANSI X3.9-1978)

Memory required to execute with typical data: 64 Kwords

No. of bits in a word: 32

Peripherals used: terminal

No. of lines in distributed program, including test run output: 507

Keywords: 3-D convolution, numerical quadrature, Gauss–Legendre, Gauss–Chebychev

Nature of physical problem
Problems in physics, chemistry, applied mathematics as well as in other fields, are often in need of an efficient 3-D convolution quadrature.

Method of solution
Algebraic manipulation and repeated use of Gaussian integration.

Correspondence to: I.E. Lagaris, Physics Department, University of Ioannina, 45110 Ioannina, Greece. E-mail: Lagaris@grioanun.

[1] E-mail: Jimmy@grioanun.

Restrictions on the complexity of the problem
Care must be taken to choose the integration cut-off properly.

Typical running time
Depending on the order of the polynomials chosen. The provided test run took 10.2 CPU seconds on the CD4680.

PROGRAM SUMMARY

Title of program: CONVUS – HIGH MEMORY

Catalogue number: ACNJ

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Licensing provisions: none

Computer for which the program is designed and others on which it has been tested:
Computers: (i) Control Data CD-4680, (ii) CRAY XMP, (iii) IBM RISC System/6000, (iv) IBM 3090; *Installation*: University of Ioannina, Ioannina, Greece

Operating system: (i) EP/IX 1.4.3, (ii) COS, (iii) A/IX, (iv) CMS

Programming language used: ANSI Fortran 77 (ANSI X3.9-1978)

Memory required to execute with typical data: 6 Mwords

No. of bits in a word: 32

Peripherals used: Terminal

No. of lines in distributed program: 478

Typical running time
Same as for the low memory version (10.2 seconds). However, if many integrals are to be calculated, the required time for each additional integral is only 1.5 seconds.

LONG WRITE-UP

1. Introduction

Integrals of the form: $Z(\boldsymbol{r}_1, \boldsymbol{r}_2) = \int d^3 r_3 \; p(\boldsymbol{r}_3)\, X(\boldsymbol{r}_1, \boldsymbol{r}_3)\, Y(\boldsymbol{r}_2, \boldsymbol{r}_3)$ can be calculated by brute force Monte Carlo quadrature or by some grid-type method. The first requires a considerable amount of computer time and hence it is appropriate only if a few integrals are to be evaluated. The second requires dense grids to achieve some level of accuracy and is thus in need of large central memory configurations, or use of the virtual memory mechanism which slows the calculation down since in that case use is made of disc related I/O operations. Here we present an efficient grid-type method that takes advantage of the high accuracy of the Gaussian quadrature [1] to reduce the number of grid points necessary for the calculation. The presented code has been used successfully for solving iteratively a system of 10 coupled integral equations that arise in variational calculations of finite nuclei [2].

2. Analysis of the problem

The X, Y and Z functions can be written in a more explicit form as $X(r_1, r_3, r_{13})$, $Y(r_2, r_3, r_{23})$ and $Z(r_1, r_2, r_{12})$ where $r_i = |\boldsymbol{r}_i|$ and $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$ and are supposed to be tabulated accordingly. (Bold letters denote vectors.) With reference to fig. 1, where the geometry is presented, one can pick a Cartesian (x, y, z) system such that the $\boldsymbol{r}_1$ vector is along the z-axis, the $\boldsymbol{r}_2$ vector is on the x–z plane

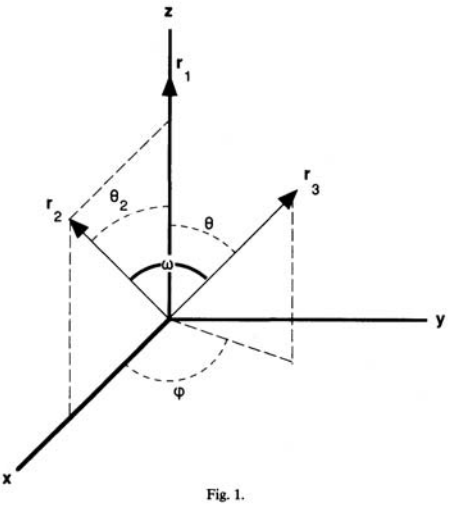

Fig. 1.

making an angle θ_2 with r_1, while the r_3 vector is defined by the (r_3, φ, θ) polar coordinates. The angle between the vectors r_2 and r_3 is ω and the following relations hold:

$$\cos\theta = \frac{r_1^2 + r_3^2 - r_{13}^2}{2r_1 r_3}, \qquad \cos\theta_2 = \frac{r_1^2 + r_2^2 - r_{12}^2}{2r_1 r_2}, \tag{2.1a, b}$$

$$\cos\omega = \frac{r_2^2 + r_3^2 - r_{23}^2}{2r_2 r_3}, \qquad \cos\omega = \cos\theta\cos\theta_2 + \sin\theta\sin\theta_2\cos\varphi. \tag{2.1c, d}$$

Rewriting the integral in terms of r_3, r_{13} and φ we get

$$Z(r_1, r_2, r_{12}) = \int_0^R r_3 p(r_3)\,\mathrm{d}r_3 \int_{|r_1 - r_3|}^{r_1 + r_3} \frac{r_{13}}{r_1} X(r_1, r_3, r_{13})\,\mathrm{d}r_{13} \int_0^{2\pi} Y(r_2, r_3, r_{23})\,\mathrm{d}\varphi, \tag{2.2}$$

where R is a cut off such that no appreciable contribution to the integral is gained by integrating further. The last integral over φ is invariant under the substitution $\varphi \to 2\pi - \varphi$ and hence its limits can go from 0 to π picking up a factor of two. Changing the integration variable from φ to r_{23} and noting that

$$\mathrm{d}\varphi = \frac{1}{\sin\theta\sin\theta_2\sin\varphi}\frac{r_{23}}{r_2 r_3}\mathrm{d}r_{23}, \tag{2.3}$$

we obtain

$$Z(r_1, r_2, r_{12}) = \frac{2}{r_1 r_2}\int_0^R \mathrm{d}r_3\; p(r_3) \int_{|r_1 - r_3|}^{r_1 + r_3} \mathrm{d}r_{13}\; r_{13} X(r_1, r_3, r_{13}) \int_a^b \mathrm{d}r_{23}\; r_{23} \frac{Y(r_2, r_3, r_{23})}{\sin\theta\sin\theta_2\sin\varphi},$$

where

$$a^2 = r_2^2 + r_3^2 - 2r_2r_3\cos(\theta - \theta_2) \quad \text{and} \quad b^2 = r_2^2 + r_3^2 - 2r_2r_3\cos(\theta + \theta_2). \tag{2.4}$$

Also note that using eq. (2.1 d) we readily get

$$\sin\theta \sin\theta_2 \sin\varphi = \sqrt{(\cos\omega - \cos(\theta + \theta_2))(\cos(\theta - \theta_2) - \cos\omega)}\,. \tag{2.5}$$

Using (2.4) and (2.1 c) the denominator of the innermost integral becomes

$$\sqrt{\frac{b^2 - r_{23}^2}{2r_2r_3}\frac{r_{23}^2 - a^2}{2r_2r_3}} = \frac{1}{2r_2r_3}\sqrt{(b^2 - r_{23}^2)(r_{23}^2 - a^2)} \tag{2.6}$$

and hence

$$Z(r_1, r_2, r_{12}) = \frac{4}{r_1}\int_0^R \mathrm{d}r_3\, r_3 p(r_3) \int_{|r_1 - r_3|}^{r_1 + r_3} \mathrm{d}r_{13}\, r_{13} X(r_1, r_3, r_{13}) \int_a^b \frac{\mathrm{d}r_{23}\, r_{23} Y(r_2, r_3, r_{23})}{\sqrt{(b^2 - r_{23}^2)(r_{23}^2 - a^2)}}.$$

Note that the last integral in r_{23} is improper since at the integration limits the denominator vanishes. There are quite a few ways to get around this problem, however, not all of them are sufficiently efficient or accurate. In what follows we describe the procedure we chose that turns out to be accurate, efficient and economical as far as computer memory is concerned.

3. Calculational details

We apply two variable transformations,

(i) $r_{13}(s) = f(r_1, r_3)s + |r_1 - r_3|$, where $f(r_1, r_3) \equiv r_1 + r_3 - |r_1 - r_3|$ with $0 < s < 1$, (3.1)

(ii) $r_{23}(t) = \frac{1}{2}(b - a)t + \frac{1}{2}(b + a)$, with $-1 < t < 1$, (3.2)

and the integral is rewritten as

$$Z(r_1, r_2, r_{12}) = \frac{4}{r_1}\int_0^R r_3\, \mathrm{d}r_3\, p(r_3)\, f(r_1, r_3) \int_0^1 \mathrm{d}s\big[f(r_1, r_3)\, s + |r_1 - r_3|\big] X(r_1, r_3, r_{13}(s))$$
$$\times \int_{-1}^1 \frac{\mathrm{d}t}{\sqrt{1 - t^2}} \frac{[(b - a)t + b + a] Y(r_2, r_3, r_{23}(t))}{\sqrt{(b - a)t + 3b + a}\sqrt{(b - a)t + b + 3a}}. \tag{3.3}$$

The last integral in t is improper since the denominator vanishes at the end points $t = \pm 1$. To avoid this problem we use the Gauss–Chebychev quadrature for the t-integral. Recall [1] that integrals of the form $W(g) = \int_{-1}^1 \mathrm{d}tg(t)/\sqrt{1 - t^2}$ can be calculated by using the Gauss–Chebychev formula $W(g) = (\pi/(n + 1))\sum_{k=0}^n g(x_k)$ with $x_k = \cos((2k + 1)\pi/2(n + 1))$ and so the $t = \pm 1$ singularity is avoided. We find that $n = 4$ (i.e. five t-grid points) is enough for most purposes.

The Gauss–Legendre quadrature is used for the s and the r_3 integrals. Note that since all of the functions X, Y, Z should be tabulated in the same way *, we chose the r-grid for the first two arguments

* This is important in the case where one has to solve integral equations iteratively and the Z function is also part of the integrand.

and the s-grid for the third argument. However, we are using values of Y on points of the t-grid in the third argument. These values of Y are not tabulated and hence we approximate them using a four-point interpolation,

$$Y(r_2, r_3, r_{23}(t)) = \sum_s c_{t,s} Y(r_2, r_3, r_{23}(s)), \tag{3.4}$$

where $c_{t,s}$ are the Lagrange interpolation coefficients. Note that if Y changes rapidly, this may introduce numerical inaccuracies. In that case one should use a denser s-grid so as to increase the accuracy of the interpolation.

The accuracy of the code depends on both, the order of the canonical polynomials used, as well as on the precision of the machine. If one uses Legendre polynomials of order up to twelve and Chebychev polynomials of order up to seven, single precision (32 bit) arithmetic is enough. Double precision arithmetic should be used if polynomials of higher order are employed.

The integration cut-off must be chosen with care. It should not be too large since in that case one may turn out integrating the function in areas where its value is negligible. A correct choice for R will distribute the evaluation points in an area where the function is substantial. Some experimentation may be necessary for picking a right value for R.

4. Description of the code

4.1. Low-memory version

The program is a collection of seven subprograms and a driver main program. In what follows we describe the organization of the program routine by routine.

4.1.1. SUBROUTINE CONSGR

This routine constructs the three different grids R(I), S(I) and T(I) and tabulates several expressions that are repeatedly used. For example the expression: $r_1 + r_2 - |r_1 - r_2|$ is tabulated in array PF(I1, I2) and the angle between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ is tabulated in array THETA(I1, I2, I12).

The Gauss–Legendre quadrature is used for the r- and s-integrations and the Gauss–Chebychev quadrature is used for the t-integration. For this reason the roots of the Legendre polynomials are tabulated along with the corresponding weights. The tabulation is performed only for the even Legendre polynomials with orders ranging from 10 to 20. The Chebychev nodes are tabulated for the polynomials with orders ranging from 5 to 10. The various orders are set by the NPI, NPIJ and NTSE parameters. (NPI and NPIJ may take on the values 10, 12, 14, 16, 18, 20, and are associated with the r- and s-grids, while NTSE may take on the values 5, 6, 7, 8, 9, 10 and is associated with the t-grid.) The Lagrange interpolation coefficients are stored in array CLAG.

4.1.2. SUBROUTINE INTE3D

Performs all the summations, collecting the appropriate terms that are involved in the integration formula.

4.1.3. FUNCTION G

Calculates the part of the t-integrand, $[(b-a)t + b + a]/(\sqrt{(b-a)t + 3b + a}\,\sqrt{(b-a)t + b + 3a})$, that needs special treatment in the case where $a = b = 0$.

4.1.4. FUNCTION LIND

Given a value at which interpolation must be performed, this routine finds the appropriate four consecutive points of the s-grid to use in the interpolation formula and returns the index of the lowest one.

4.1.5. BLOCK DATA INTEG

Stores the Legendre polynomial zeros, the associated integration weights, as well as the Gauss–Chebychev nodes. The program is coded in single precision, and it can be easily modified via IMPLICIT DOUBLE PRECISION (A–H, O–Z) statements at the top of each routine. However, the above constants are stored using 14 significant digits, which may cause some compilers to issue warning messages. In that case one may manually truncate the excess digits.

4.1.6. FUNCTION CHECK

Checks the values of the parameters NPI, NPIJ, NTSE in case they are erroneously modified by the user.

4.1.7. PROGRAM CONVUS

First the cut-off radius is input and the CONSGR routine is called to evaluate and store all the necessary quantities. Then one constructs the X13 and Y23 arrays accordingly. As an example we programmed the degenerate case with $p(r_3) = 1$, $X(r_1, r_3, r_{13}) = \exp(-r_{13}^2)$ and $Y(r_2, r_3, r_{23}) = \exp(-r_{23}^2)$ which can be integrated analytically and yields $(\frac{1}{2}\pi)^{3/2} \exp(-\frac{1}{2}r_{12}^2)$, a result that can be used for comparison. Note that r_{13} is calculated by means of eq. (3.1)

4.1.8. FUNCTION FUN

Contains the code for the analytic result of the above example, and is used for comparison.

4.2. High-memory version

This version is quite memory-demanding and it is not any faster than the low-memory version if only one integral is to be evaluated. However, if many integrals are to be calculated, then the computational cost for each additional integral evaluation, is only a fraction ($\sim \frac{1}{7}$) of the low-memory version.

The following two routines are different in the high-memory version and are subsequently described.

4.2.1. SUBROUTINE CONSGR

The labeled common block /CRAY/ holding two arrays is added. Array DU(K, I23, I13, I12, I3, I2, I1) stores for every point (r_1, r_2, r_{12}) the integrand except for the $X(r_1, r_3, r_{13})$ and $Y(r_2, r_3, r_{23})$ values. Array ID(J, I13, I12, I3, I2, I1) stores the appropriate interpolation indices.

4.2.2. SUBROUTINE INTE3D

Performs a much simpler summation, since the integrand is already tabulated for the most part.

5. Description of the test run

The only input required is the cut-off limit R. In the test run $R = 4$ is used. The following two commands were used to compile and execute the program (assuming the code resides in file convus.f):

```
f77 -O2 -o convus convus.f
convus
```

References

[1] A.H. Stroud and D. Secrest, Gaussian Quadrature Formulas (Prentice-Hall, Englewood Cliffs, NJ, 1966).
[2] G. Co, A. Fabrocini, S. Fantoni and I.E. Lagaris, Nucl. Phys. A549 (1992) 439.

TEST RUN OUTPUT

NUMERICAL	ANALYTIC	DIFFERENCE
1.96868E+00	1.96870E+00	2.11000E-05
1.96764E+00	1.96768E+00	3.52859E-05
1.96384E+00	1.96385E+00	1.40667E-05
9.26851E-01	9.26087E-01	7.64430E-04
8.92178E-01	8.90412E-01	1.76662E-03
8.49116E-01	8.48223E-01	8.93772E-04
1.41850E-02	1.42997E-02	1.14634E-04
1.28337E-02	1.29439E-02	1.10168E-04
1.13739E-02	1.14681E-02	9.41390E-05
9.26085E-01	9.26087E-01	1.43051E-06
8.90407E-01	8.90412E-01	4.29153E-06
8.48195E-01	8.48223E-01	2.75373E-05
1.96842E+00	1.96817E+00	2.58088E-04
1.07030E+00	1.06797E+00	2.33781E-03
1.08525E-01	1.08552E-01	2.76640E-05
3.03663E-01	3.03597E-01	6.64890E-05
2.02420E-02	2.08239E-02	5.81907E-04
1.70486E-04	1.76166E-04	5.67932E-06
1.42997E-02	1.42997E-02	8.38190E-09
1.29439E-02	1.29439E-02	0.00000E+00
1.14669E-02	1.14681E-02	1.15763E-06
3.03532E-01	3.03597E-01	6.52075E-05
2.06826E-02	2.08239E-02	1.41229E-04
1.75106E-04	1.76166E-04	1.06001E-06
1.83955E+00	1.96533E+00	1.25783E-01
3.60471E-02	4.16409E-02	5.59377E-03
9.31165E-09	2.28822E-08	1.35705E-08