The American Society of Mechanical Engineers

Reprinted From
Symbolic Computation in Fluid Mechanics and Heat Transfer — HTD-Vol. 105/AMD-Vol. 97
Editors: H. H. Bau, T. Herbert, and M. M. Yovanovich J. R. Stenner, and W. J. Bryan
(Book No. G00483)

THE USE OF SYMBOLIC COMPUTATION IN PERTURBATION ANALYSIS

R. H. Rand
Department of Theoretical and Applied Mechanics
Cornell University
Ithaca, New York

Abstract

MACSYMA programs are presented which automate i) the method of composite expansions for treating linear variable coefficient second order ordinary differential equation boundary value problems, and ii) the asymptotic expansion of a class of definite integrals.

Introduction

Perturbation methods (also known as asymptotic expansions) are a diverse class of approximate but algebraic (as opposed to numerical) methods for treating problems in applied mathematics. Although these methods are so varied to prohibit a general definition, most involve problems which contain a small parameter, and consist of a series expansion asymptotically valid in the limit as the small parameter goes to zero.

The computational procedure for perturbation methods invariably involves tedious quantities of algebra, especially if the series expansion is carried to more than just one or two terms. These computations have become less burdensome and more accurately and efficiently executed since the recent availability of computer algebra (also known as symbolic manipulation) software. These systems provide computer environments which permit the usual processes of algebra and calculus to be performed on unevaluated symbolic variables (i.e., letters as opposed to numbers).

In two previous research monographs [4,5], the author has investigated the use of the computer algebra system MACSYMA to automate the following perturbation methods: Lindstedt's method, center manifold reduction, normal forms, averaging, two variable expansion method (also known as multiple scales), Lie transforms and Liapunov-Schmidt reduction. The present paper extends this work by presenting MACSYMA programs which implement i) the method of composite expansions and ii) the asymptotic expansion of integrals. For both of these we describe the method and offer an example completed by hand. Then we give a sample run of the computer algebra program and the program listing.

The reader may obtain an electronic copy of these programs (as well as all the programs in [5]), by contacting the author via bitnet at RHRY@CRNLVAX5.

The Method of Composite Expansions

This method is an alternative to matched aymptotic expansions for differential equations of the form:

(A1) $$\epsilon\, y'' + a(x)\, y' + b(x)\, y = 0$$

with two boundary conditions:

(A2) $$y(0) = \alpha, \quad y(1) = \beta,$$

where primes represent differentiation with respect to x. In (A1), $\epsilon << 1$ and a(x) and b(x) are analytic functions of x on [0,1]. We assume that $a(x) > 0$ throughout [0,1], in which case a boundary layer (b.l.) will occur at x = 0. (In the opposite case that $a(x) < 0$, the b.l. will occur at x = 1, but changing independent variable to $\xi = 1-x$ will bring the b.l. to $\xi = 0$. We assume that a(x) does not change sign in [0,1], since this case leads to more complicated behavior (internal b.l.'s), see [2,3].)

The following description of the method is based on Nayfeh [3], pp.148-149: We look for a solution to (A1) and (A2) in the form:

(A3) $$y = \sum_{n=0}^{\infty} \epsilon^n f_n(x) + e^{-g(x)/\epsilon} \sum_{n=0}^{\infty} \epsilon^n h_n(x)$$

in which the functions f_n, g and h_n are to be found. We require $g(x) \sim x$ in the limit $x \to 0$, since the b.l. is known to have thickness x. The procedure is to substitute (A3) into (A1) and (A2), collect terms, and equate the coefficients of ϵ^n and $\epsilon^n e^{-g/\epsilon}$ to zero. We proceed directly to an example [3]:

(A4) $$\epsilon\, y'' + (2x+1)\, y' + 2\, y = 0$$

for which a(x) = 2x+1 and b(x) = 2. Substituting (A3) into (A4) and multiplying by ϵ gives the following lowest order terms:

(A5) $e^{-g/\epsilon}$: $\quad g'\,h_0\,(g' - 2x - 1) = 0$

(A6) $\epsilon\, e^{-g/\epsilon}$: $\quad h_0\,(2 - g'') + h_0'\,(2x + 1 - 2g') = 0$

(A7) ϵ: $\quad 2 f_0 + (2x + 1)\, f_0' = 0$

Substituting (A3) into the b.c. (A2) gives:

(A8) $\quad f_0(0) + h_0(0) = \alpha \quad$ and $\quad f_0(1) = \beta$

Eq.(A5) gives either $h_0 = 0$, g = constant (both of which must be rejected), or:

(A9) $$g(x) = x^2 + x$$

in which the arbirary constant of integration has been taken as zero in order that $g(x) \sim x$. Substituting (A9) into (A6) gives

(A10) $\quad h_0' = 0 \quad$ or $\quad h_0(x)$ = constant.

Next, (A7) may be integrated to give:

(A11) $$f_0(x) = \frac{\text{constant}}{2x + 1}.$$

The arbitrary constants of integration in (A10) and (A11) are found by using the b.c.(A8):

(A12) $$f_0(x) = \frac{3\beta}{2x + 1}, \quad h_0(x) = \alpha - 3\beta$$

Substituting (A9) and (A12) into (A3) gives the approximate result:

(A13) $$y = \frac{3\beta}{2x + 1} + (\alpha - 3\beta)\, e^{-\frac{x^2+x}{\epsilon}} + O(\epsilon)$$

This type of computation may be automated using computer algebra. The MACSYMA program "composite" which we present here accomplishes this task for general functions a(x) and b(x), to arbitrary order of truncation. Here is a sample run on the previous example:

```
composite();
The d.e. is: ey''+a(x)y'+b(x)y=0
with b.c. y(0)=y0 and y(1)=y1
enter a(x) > 0 on [0,1]
2*x+1;
enter b(x)
2;
enter y0
alpha;
enter y1
beta;

The d.e. is: ey''+( 2 x + 1 )y'+( 2 )y=0
with b.c. y(0)= alpha and y(1)= beta

enter truncation order
3;
```

```
                       3              2
       85312 beta e    928 beta e    16 beta e
  (- ------------- - ----------- - ---------
           243            27            3

                        2
                       x  + x
                     - ------
                         e
- 3 beta + alpha) %e

  3             6                 5                 4
- e  (5120 beta x  + 15360 beta x  + 21504 beta x

                  3                2
+ 17408 beta x  + 16032 beta x  + 9888 beta x

                          7            6            5
- 85312 beta)/(31104 x  + 108864 x  + 163296 x

            4          3          2
+ 136080 x  + 68040 x  + 20412 x  + 3402 x + 243)

  2              4               3               2
- e  (128 beta x  + 256 beta x  + 336 beta x

                                     5          4          3
+ 208 beta x - 928 beta)/(864 x  + 2160 x  + 2160 x

           2
+ 1080 x  + 270 x + 27)

                 2
   e (8 beta x  + 8 beta x - 16 beta)    3 beta
 - ---------------------------------- + -------
               3        2                2 x + 1
           24 x  + 36 x  + 18 x + 3
```

[VAX 8530 computation time = 112 sec.]

In order to check the accuracy of the results, we used finite differences to obtain a numerical solution to (A4),(A2) for $\alpha = 1$, $\beta = 2$. The following table lists the values of y'(0) obtained from various truncations of the composite expansion, along with the finite difference value, for ϵ = 0.01,0.05 and 0.09:

<u>Table of approximate values of y'(0)</u>

truncation order	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.09$
0	488	88.0	43.55
1	497.97	95.2	47.98
2	498.59	96.88	48.48
3	498.65	97.40	46.94
finite diff.	498.66	98.64	54.16

Note that while additional accuracy is obtained by taking more terms in the expansion for $\epsilon = 0.01$ and $\epsilon = 0.05$, this is no longer true for $\epsilon = 0.09$. This behavior is typical of asymptotic series (as opposed tc convergent series.) The two limiting processes of letting $\epsilon \to 0$ and truncating the series after $N \to \infty$ terms are in competition. In an asymptotic series, we fix N and let $\epsilon \to 0$, while in a convergent series, we fix ϵ and let $N \to \infty$. In terms of the Table, if we fix the truncation order at N = 3 and take ϵ smaller, the difference between the asymptotic series approximation and the exact solution (represented by the finite differences result) becomes smaller. If we fix ϵ, however, we cannot expect that increasing the order of truncation N will necessarily increase the accuracy of the approximation, cf. $\epsilon = 0.09$.

Here is the program listing:

```
/* method of composite expansions */
composite():=(
/* input problem from keyboard */
print("The d.e. is: ey''+a(x)y'+b(x)y=0"),
print("with b.c. y(0)=y0 and y(1)=y1"),
a:read("enter a(x) > 0 on [0,1]"),
b:read("enter b(x)"),
y0:read("enter y0"),
y1:read("enter y1"),
print("The d.e. is: ey''+(",a,")y'+(",b,")y=0"),
print("with b.c. y(0)=",y0,"and y(1)=",y1),
trunc:read("enter truncation order"),
/* set up basic form of solution */
y:sum(f[n](x)*e^n,n,0,trunc)+
  %e^(-g(x)/e)*sum(h[n](x)*e^n,n,0,trunc),
/* substitute into d.e. */
de1:e*diff(y,x,2)+a*diff(y,x)+b*y,
/* expand and collect terms */
de2:expand(e*de1),
gstuff:coeff(de2,%e^(-g(x)/e)),
other:expand(de2-gstuff*%e^(-g(x)/e)),
gstuff2:taylor(gstuff,e,0,trunc+1),
other2:taylor(other,e,0,trunc+1),
for i:0 thru trunc+1 do
   geq[i]:coeff(gstuff2,e,i),
for i:1 thru trunc+1 do
   otheq[i]:coeff(other2,e,i),
/* find g(x) */
geq:geq[0]/h[0](x)/diff(g(x),x),
gsol:ode2(geq,g(x),x),
resultlist:[g(x)=ev(rhs(gsol),%c=0)],
/* find h[i](x)'s */
/* hh[i] = constant of integration */
for i:0 thru trunc do(
   heq[i]:ev(geq[i+1],resultlist,diff),
   hsol[i]:ode2(heq[i],h[i](x),x),
   hsol[i]:ev(hsol[i],%c=hh[i]),
   resultlist:append(resultlist,[hsol[i]])),
/* find f[i](x)'s */
/* ff[i] = constant of integration */
for i:0 thru trunc do(
   feq[i]:ev(otheq[i+1],resultlist,diff),
   fsol[i]:ode2(feq[i],f[i](x),x),
   fsol[i]:ev(fsol[i],%c=ff[i]),
   resultlist:append(resultlist,[fsol[i]])),
/* boundary conditions */
/* y(0)=y0 and y(1)=y1 */
bc1:ev(y=y0,resultlist,x=0),
bc2:ev(y=y1,resultlist,x=1),
/* kill exponentially small %e terms */
bc2:subst(0,%e,bc2),
for i:0 thru trunc do
   (bc1[i]:ratcoef(bc1,e,i),
    bc2[i]:ratcoef(bc2,e,i)),
/* solve for unknown consts */
bceqs:makelist(bc1[i],i,0,trunc),
bceqs:append(bceqs,makelist(bc2[i],i,0,trunc)),
bcunk:makelist(hh[i],i,0,trunc),
bcunk:append(bcunk,makelist(ff[i],i,0,trunc)),
const:solve(bceqs,bcunk),
/* substitute back */
resultlist:ratsimp(ev(resultlist,const)),
ysol:ev(y,resultlist))$
```

Asymptotic Expansion of Integrals

This procedure is concerned with generating approximations for definite integrals of the form

$$\text{(B1)} \qquad \int_a^b f(t)\, e^{x\varphi(t)}\, dt$$

in the limit as $x \to \infty$. The following description of the method follows Bender and Orszag [2], Chapter 6.

The idea of the method is that $e^{x\varphi(t)}$ makes its largest contribution to the integral in the neighborhood of the point $t = c$ at which $\varphi(t)$ is maximum. In the large x limit, we may (with asymptotically small error) replace (B1) by

$$\text{(B2)} \qquad \int_{c-\epsilon}^{c+\epsilon} f(t)\, e^{x\varphi(t)}\, dt, \quad \text{if } a < c < b,$$

or by

$$\text{(B3)} \qquad \int_{a}^{a+\epsilon} f(t)\, e^{x\varphi(t)}\, dt, \quad \text{if } c = a,$$

or by

$$\text{(B4)} \qquad \int_{b-\epsilon}^{b} f(t)\, e^{x\varphi(t)}\, dt, \quad \text{if } c = b.$$

Then we may expand the integrand in a truncated Taylor series about $t = c$. In order to evaluate the resulting integrals most easily, we next let $\epsilon \to \infty$, i.e., we respectively replace the integrals

$$\int_{c-\epsilon}^{c+\epsilon}, \int_{a}^{a+\epsilon}, \int_{b-\epsilon}^{b} \text{ in (B2)-(B4) by } \int_{-\infty}^{\infty}, \int_{a}^{\infty}, \int_{-\infty}^{b}.$$

As an example, we take the following expression for the modified Bessel function of the first kind ([2],p.270):

$$\text{(B5)} \qquad I_n(x) = \frac{1}{\pi}\int_0^{\pi} \cos(nt)\, e^{x\cos(t)}\, dt$$

Here $\varphi(t) = \cos(t)$ achieves its maximum at $c = 0$ so that

$$\text{(B6)} \qquad I_n(x) \sim \frac{1}{\pi}\int_0^{\epsilon} \cos(nt)\, e^{x\cos(t)}\, dt$$

Expanding cos(t) in a Taylor series around $t = c = 0$,

$$\text{(B7)} \qquad e^{x\cos(t)} = e^{x}\left(1 - \frac{t^2}{2} + \frac{t^4}{24} - \cdots\right)$$

In (B7), the leading term, e^x, is independent of t and can be passed outside of the integral (B6). The next term, $e^{-xt^2/2}$, is the key factor in the evaluation of the integral (B7). Expanding all the other factors in the integrand of (B7) in Taylor series, we obtain:

(B8) $$I_n(x) \sim \frac{e^x}{\pi}\int_0^{\epsilon}\left(1-\frac{n^2t^2}{2}+\cdots\right)e^{-xt^2/2}\left(1+\frac{xt^4}{24}+\cdots\right)dt$$

Now we let $\epsilon \to \infty$ (thereby adding terms to (B8) with asymptotically small value), and set $t = s/\sqrt{x}$, in order to evaluate the resulting integrals:

(B9) $$I_n(x) \sim \frac{e^x}{\pi}\int_0^{\infty}\left(1-\frac{n^2s^2}{2x}+\cdots\right)e^{-s^2/2}\left(1+\frac{s^4}{24\,x}+\cdots\right)\frac{ds}{\sqrt{x}}$$

$$\sim \frac{e^x}{\pi\sqrt{x}}\left[\int_0^{\infty}e^{-s^2/2}\,ds+\frac{1}{x}\int_0^{\infty}e^{-s^2/2}\left[\frac{s^4}{24}-\frac{n^2s^2}{2}\right]ds\right] + O(x^{-2})$$

The integrals in (B9) may be evaluated to give:

(B10) $$I_n(x) \sim \frac{e^x}{\sqrt{2\pi x}}\left[1+\frac{1}{x}\left[\frac{1}{8}-\frac{n^2}{2}\right]+O(x^{-2})\right]$$

In order to obtain higher order terms in this asymptotic expansion, we need to keep additional terms in the truncated Taylor series in eq.(B8). The number of terms to be kept depends upon the key factor in $e^x\varphi(t)$. We find that if the key factor is e^{xku^2} (where $u = t - c$ and where k is some constant), as it is in the foregoing example, then 6n terms must be maintained in the Taylor series truncations to account for terms of $O(x^{-n})$. In the case that the key factor is e^{xku}, we find that 2n terms in the truncations in t account for terms of $O(x^{-n})$ in the result. Truncation points for other key factors, e.g. e^{xku^3}, are not known.

This type of computation may be automated using computer algebra. The MACSYMA program "asymptotic" which we present here accomplishes this task for general functions f(t) and $\varphi(t)$, to arbitrary order of truncation. (We require that $\varphi(t)$ behave like (t-c) or $(t-c)^2$ near its maximum t = c, since we know truncation points only for these key factors.) Here is a sample run on the previous example:

```
asymptotic();
The integrand is of the form: f(t) exp(x phi(t))
enter f(t)
cos(n*t);
enter phi(t)
cos(t);
enter the lower limit of integration
0;
enter the upper limit of integration
%pi;
                                  cos(t) x
The integrand is cos(n t) %e

integrated from 0 to %pi
enter value of t at which phi = cos(t)  is maximum
0;
enter truncation order
4;
```

```
                                    4          2  3
     sqrt(2) sqrt(%pi) (98304 x  - 49152 n  x

          3          4  2          2  2          2
 + 12288 x  + 12288 n  x  - 30720 n  x  + 6912 x

          6          4          2
 - 2048 n  x + 17920 n  x - 33152 n  x + 7200 x

          8          6          4          2          x
 + 256 n  - 5376 n  + 31584 n  - 51664 n  + 11025) %e

          9/2
/(196608 x   )
```

[VAX 8530 computation time = 309 sec.]

This result agrees with [1], p.377, eq.(9.7.1).

Before giving the program listing, we offer another example, Stirling's formula ([2], p.275-6). The gamma function $\Gamma(x)$ is given by the integral

(B11) $$\Gamma(x) = \int_0^{\infty}e^{-t}\,t^{x-1}\,dt = \int_0^{\infty}\frac{1}{t}\,e^{-t+x\,\ln(t)}\,dt$$

Eq.(B11) is not of the form (B1). However, by setting $\tau = t/x$, (B11) takes the form

(B12) $$\Gamma(x) = x^x\int_0^{\infty}\frac{1}{\tau}\,e^{x(\ln(\tau)-\tau)}\,d\tau$$

In order to apply the program "asymptotic" to this example, we note that $\varphi(\tau) = \ln(\tau)-\tau$ achieves its maximum value at $\tau = 1$. Here is the run:

```
asymptotic();
The integrand is of the form: f(t) exp(x phi(t))
enter f(t)
1/t;
enter phi(t)
log(t)-t;
enter the lower limit of integration
0;
enter the upper limit of integration
inf;
                                (log(t) - t) x
                              %e
The integrand is ----------------
                                    t

integrated from 0 to inf
enter value of t at which phi = log(t) - t  is maximum
1;
enter truncation order
4;
```

```
                                      4          3
     sqrt(2) sqrt(%pi) (2488320 x  + 207360 x

                2                       - x          9/2
       + 8640 x  - 6672 x - 571) %e    /(2488320 x    )
```

[VAX 8530 computation time = 206 sec.]

These results agree with [1], p.257, eq.(6.1.37). Numerical evaluation of these results indicates the advantage of taking additional terms in the expansion.

From (B12), we multiply the results by x^x and compare with (x-1)!. For x = 100, MACSYMA gives the following exact value for 99!:

```
933262154439441526816992388562667004907159682643816214 6
859296389521759999322991560894146397615651828625369792 0
827223758251185210916864000000000000000000000000
```

Numerical evaluation for x = 100

no.terms	value + 10^{155}	no.correct digits
1	9.32484762526934324776475612718	2
2	9.33261833162373436713789342395	6
3	9.33262156941804869677096556449	8
4	9.33262154441508149166991017407	10
5	9.33262154439368356859719928884	12

Here is the program listing:

```
/* asymptotic expansion of integrals */
asymptotic():=(
block(
/* input the problem from the keyboard */
print("The integrand is of the form:
       f(t) exp(x phi(t))"),
f:read("enter f(t)"),
phi:read("enter phi(t)"),
a:read("enter the lower limit of integration"),
b:read("enter the upper limit of integration"),
print("The integrand is",f*%e^(x*phi)),
print("integrated from",a,"to",b),
c:read("enter value of t at which phi =",phi,
       " is maximum"),
/* set up limits of integration for later use */
if c=a then (lowerlim:0, upperlim:inf)
   else if c=b then (lowerlim:minf, upperlim:0)
       else (lowerlim:minf, upperlim:inf),
/* move origin to t=c with u=t-c */
f1:ev(f,t=u+c),
phi1:ev(phi,t=u+c),
trunc:read("enter truncation order"),
/* determine lowest non-constant term in taylor series
   for phi about u=0 */
phi2:taylor(phi1,u,0,2),
phic:ev(phi,t=c),
keypower:lopow(phi2-phic,u),
if keypower=1 then trunc1:2*trunc else
   if keypower=2 then trunc1:6*trunc else
      (print("phi does not behave like t-",c,
             "or (t-",c,")^2 near t=",c),
       return("program aborted")),
phi3:taylor(phi1,u,0,trunc1),
phikey:coeff(phi3,u,keypower)*u^keypower,
phirest:phi3-phikey-phic,
/* set flag so integration routine assumes x is
positive */
assume_pos:true,
integrand:taylor(f1*%e^(x*phirest),u,0,trunc1),
/* set u = s/x^(1/keypower) */
integrand2:ev(integrand,u=s/x^(1/keypower)),
/* set x = 1/xx in order to truncate */
integrand3:taylor(ev(integrand2,x=1/xx),xx,0,trunc),
/* return to x again */
integrand4:ev(integrand3,xx=1/x),
approx:%e^(x*phic)/x^(1/keypower)*
       integrate(integrand4*%e^(ev(phikey,u=s)),s,
                 lowerlim,upperlim),
approx2:factor(approx)))$
```

Acknowledgement

This work was partially supported by the Mathematical Sciences Institute at Cornell University and by the Air Force Office of Scientific Research.

References

1. Abramowitz,M. and Stegun,I.A., "Handbook of Mathematical Functions", 1046 pp., Dover (1965)
2. Bender,C.M. and Orszag,S.A., "Advanced Mathematical Methods for Scientists and Engineers", 593 pp., McGraw-Hill (1978)
3. Nayfeh,A., "Perturbation Methods", 425 pp., Wiley (1973)
4. Rand,R.H., "Computer Algebra in Applied Mathematics: An Introduction to MACSYMA", 181 pp., Pitman (1984)
5. Rand,R.H. and Armbruster,D., "Perturbation Methods, Bifurcation Theory and Computer Algebra", 243 pp., Springer (1987)