ALGORITHM 9 RUNGE-KUTTA INTEGRATION
3 (May 1960), 318 P. NAUR

```
procedure RK(x,y,n,FKT,eps,eta,xE,yE,fi) ; value x,y ;
integer n ; Boolean fi ; real x,eps,eta,xE ; array
y,yE ; procedure FKT ;
comment : RK integrates the system yk'=fk(x,y1,y2,..., yn)
(k=1,2,...,n) of differential equations with the method of Runge-
Kutta with automatic search for appropriate length of integration
step. Parameters are: The initial values x and y[k] for x and the un-
known functions yk(x). The order n of the system. The procedure
FKT(x,y,n,z) which represents the system to be integrated, i.e.
the set of functions fk . The tolerance values eps and eta which
govern the accuracy of the numerical integration. The end of the
integration interval xE. The output parameter yE which repre-
sents the solution at x=xE. The Boolean variable fi, which must
always be given the value true for an isolated or first entry into
RK. If however the functions y must be available at several mesh-
points x0 , x1 , ... , xn , then the procedure must be called repeat-
edly (with x=xk , xE=xk+1, for k=0, 1, ... , n-1) and then the
later calls may occur with fi=false which saves computing time.
The input parameters of FKT must be x,y,n, the output parameter
z represents the set of derivatives z[k]=fk(x,y[1], y[2], ... , y[n])
for x and the actual y's. A procedure comp enters as a non-local
identifier ;
begin
  array z,y1,y2,y3[1:n] ; real x1,x2,x3,H ; Boolean out ;
  integer k,j ; own real s,Hs ;
  procedure RK1ST(x,y,h,xe,ye) ; real x,h,xe ; array
          y,ye ;
    comment : RK1ST integrates one single RUNGE-KUTTA
          with initial values x,y[k] which yields the output
          parameters xe=x+h and ye[k], the latter being the
          solution at xe. Important: the parameters n, FKT, z
          enter RK1ST as nonlocal entities ;
    begin
      array w[1:n], a[1:5] ; integer k,j ;
      a[1] := a[2] := a[5] := h/2 ; a[3] := a[4] := h ;
      xe := x ;
      for k := 1 step 1 until n do ye[k] := w[k] := y[k] ;
      for j := 1 step 1 until 4 do
      begin
        FKT(xe,w,n,z) ;
        xe := x+a[j] ;
        for k := 1 step 1 until n do
        begin
          w[k] := y[k]+a[j]×z[k] ;
          ye[k] := ye[k] + a[j+1]×z[k]/3
        end k
      end j
    end RK1ST ;
Begin of program:
      if fi then begin H := xE-x ; s := 0 end else H := Hs ;
      out := false ;
AA:   if (x+2.01×H-xE>0)≡(H>0) then
      begin Hs := H ; out := true ; H := (xE-x)/2
        end if ;
      RK1ST (x,y,2×H,x1,y1) ;
BB:   RK1ST (x,y,H,x2,y2) ; RK1ST(x2,y2,H,x3,y3) ;
      for k := 1 step 1 until n do
          if comp(y1[k],y3[k],eta)>eps then go to CC ;
      comment : comp(a,b,c) is a function designator, the value
        of which is the absolute value of the difference of the
        mantissae of a and b, after the exponents of these
        quantities have been made equal to the largest of the ex-
        ponents of the originally given parameters a,b,c ;
      x := x3 ; if out then go to DD ;
      for k := 1 step 1 until n do y[k] := y3[k] ;
      if s=5 then begin s := 0 ; H := 2×H end if ;
      s := s+1 ; go to AA ;
CC:   H := 0.5×H ; out := false ; x1 := x2 ;
      for k := 1 step 1 until n do y1[k] := y2[k] ;
      go to BB ;
DD:   for k := 1 step 1 until n do yE[k] := y3[k]
end RK
```

[8] This RK-program contains some new ideas which are related to ideas of S. GILL, A process for the step-by-step integration of differential equations in an automatic computing machine, *Proc. Camb. Phil. Soc. Vol. 47* (1951) p. 96; and E. FRÖBERG, On the solution of ordinary differential equations with digital computing machines, *Fysiograf. Sällsk. Lund, Förhd. 20* Nr. 11 (1950) p. 136–152. It must be clear, however, that with respect to computing time and round-off errors it may not be optimal, nor has it actually been tested on a computer.

CERTIFICATION OF ALGORITHM 9 [D2] RUNGE-KUTTA INTEGRATION [P. Naur et al., *Comm. ACM 3* (May 1960), 318]

HENRY C. THACHER, JR. (Recd. 28 July 1964 and 22 Nov. 1965)
Argonne National Laboratory, Argonne, Ill.

Algorithm 9 was transcribed into the hardware representation for CDC 3600 ALGOL and run successfully. The following procedure was used for the global procedure *comp*:

```
real procedure comp (a, b, c); value a, b, c; real a, b, c;
begin integer AE, BE, CE;
  integer procedure expon(x); real x;
  comment This function produces the base 10 exponent of x;
  expon := if x = 0 then -999 else
    entier (.4342944819 × ln(abs(x)) + 1);
  comment The number -999 may be replaced by any number
    less than the exponent of the smallest positive number handled
    by the particular machine used, for this algorithm assumes
    that true zero has an exponent smaller than any nonzero
    floating-point number. Users implementing real procedure
    comp by machine code should make sure that this condition
    is satisfied by their program;
  AE := expon(a); BE := expon(b); CE := expon(c);
  if AE < BE then AE := BE; if AE < CE then AE := CE;
  comp := abs(a - b)/10 ↑ AE
end
```

This has the advantage of machine independence, but is highly inefficient compared to machine code.

The procedure was tested using the two following procedures for *FKT*:

procedure *FKT* (*X*, *Y*, *N*, *Z*); **real** *X*; **integer** *N*; **array** *Y*, *Z*;
comment $(dy_1/dx) = z_1 = y_2$, $(dy_2/dx) = z_2 = -y_1$. With $y_1(0) = 0$, $y_2(0) = 1$, the solution is $y_1 = sin\ x$, $y_2 = cos\ x$;
begin *Z* [1] := *Y* [2]; *Z* [2] := −*Y* [1] **end**;
procedure *FKT* (*X*, *Y*, *N*, *Z*); **real** *X*; **integer** *N*; **array** *Y*, *Z*;
comment $(dy_1/dx) = 1 + y_1^2$. For $y_1(0) = 0$, $y(x) = tan\ x$;
Z [1] := 1+*Y*[1]↑2;

The *RK* procedure was used to integrate the differential equations represented by the first *FKT* procedure from $x = 0(0.5)7.0$, with $eps = eta = 10^{-6}$, and with $y_1(0) = 0$, $y_2(0) = 1$. The actual step size *h* was .0625 for most of the range, but was reduced to .03125 in the neighborhood of $x = k\pi/2$, where one or the other of the solutions is small.

The computed solutions at $x = 7.0$ were: $y_1 = 6.5698602746 \times 10^{-1}$, $y_2 = 7.5390270246 \times 10^{-1}$, with errors -5.71×10^{-7} and 4.48×10^{-7}, respectively.

Results for the second differential equation are summarized in Table I below.

The efficiency of the procedure would be increased slightly on most computers by changing the type of the **own** variable *s* from **real** to **integer.**

The error is estimated by comparing the results of successive pairs of steps with that of a single double step. This is somewhat more time-consuming than the Kutta-Merson process presented in Algorithm 218 [*Comm. ACM 6* (Dec. 1963) 737–8]. However, the criterion for step-size variation in Algorithm 9 which effectively applies an approximate relative error criterion, *eps*, foı $|y| > eta$, and an absolute error criterion $eta \times eps$, for $|y| < eta$, appears superior when the solution fluctuates in magnitude.

TABLE I [ALG. 9]

		$x = 0.5$			$x = 1.0$			$x = 1.5$		
	η	h_{min}	*Absolute error*	*Relative error*	h_{min}	*Absolute error*	*Relative error*	h_{min}	*Absolute error*	*Relative error*
10^{-7}	10^{-3}	.03125	-1×10^{-9}	-2×10^{-9}	.03125	9×10^{-8}	6×10^{-8}	.00390625	-1×10^{-6}	-8×10^{-8}
10^{-5}	10^{-3}	.125	-5×10^{-7}	-9×10^{-7}	.0625	8×10^{-7}	5×10^{-7}	.0078125	-2×10^{-4}	-1×10^{-5}
10^{-3}	10^{-3}	.25	-1×10^{-5}	-2×10^{-5}	.25	-2×10^{-4}	-1×10^{-4}	.03125	-3×10^{-2}	-2×10^{-3}