stichting
mathematisch
centrum

MC

REKENAFDELING MR 137/72 AUGUST

C.J. ROOTHART AND H. FIOLET

QUADRATURE PROCEDURES

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

1.0. Introduction.

In this paper five ALGOL 60 procedures are presented, suitable for integration of definite integrals.
These quadrature procedures are selected from a total of fifteen either because they are obvious improvements of other procedures or because of better results in a few preliminary tests.
The five procedures are:

qad	(due to Schönfeld and Argelo)
int	(due to Kruseman Aretz)
integral	(due to van Emden)
qadrat	(due to Roothart)
trapex	(due to Bulirsch and Stoer).

Only qadrat is newly written, the other procedures are of earlier date.

In the next two sections the complete ALGOL 60 texts and a short description of the procedures are given. In section 4. a number of numerical tests are reported, the results are shown in graphical form. If an integral to be computed resembles one of the tested integrals then one can make use of the figures to choose the best procedure. Hereby one might attend to smoothness, the presence of singularities, discontinueties, or peaks and, of course the required precision is of importance.
In section 5 the irregularities in the figures are explained.
We are indebted to the members of the working group quadrature for many valuable suggestions.

In particular we wish to thank Dr. P.J. van der Houwen for carefully reading the manuscript and Mr. A.C. IJsselstein for writing the plotting-program by which the figures 1 - 26 were obtained.

Contents

1.0. Introduction.

In this paper five ALGOL 60 procedures are presented,suitable for integration of definite integrals.
These quadrature procedures are selected from a total of fifteen either because they are obvious improvements of other procedures or because of better results in a few preliminary tests.
The five procedures are:

qad
int
integral
qadrat
trapex.

Only qadrat is newly written, the other procedures are of earlier date.

In the next two sections the complete ALGOL 60 texts and a short description of the procedures are given. In section 4. a number of numerical tests are reported, the results are shown in graphical form. If an integral to be computed resembles one of the tested integrals then one can make use of the figures to choose the best procedure. Hereby one might attend to smoothness, the presence of singularities, discontinueties, or peaks and, of course the required precision is of importance.
In section 5 the irregularities in the figures are explained.
We are indebted to the members of the working group quadrature for many valuable suggestions.

In particular we wish to thank Dr. P.J. van der Houwen for carefully reading the manuscript and Mr. A.C. IJsselstein for writing the plotting-program by which the figures 1 - 26 were obtained.

2. Procedures

```
real procedure qad(x, a, b, fx, e); value a, b; real x, a, b, fx;
array e;
begin real x0, x1, x2, f0, f1, f2, t, v, sum, hmin, re, ae;

   procedure int;
   begin real x3, x4, f3, f4, h;
      x4:= x2; x2:= x1; f4:= f2; f2:= f1;
   anew: x:= x1:= (x0 + x2) × .5; f1:= fx;
      x:= x3:= (x2 + x4) × .5; f3:= fx; h:= x4 - x0;
      v:= (4 × (f1 + f3) + 2 × f2 + f0 + f4) × 15;
      t:= 6 × f2 - 4 × (f1 + f3) + f0 + f4;
      if abs(t) < abs(v) × re + ae then sum:= sum + h × (v - t)
      else if abs(h) < hmin then e[3]:= e[3] + 1 else
      begin int; x2:= x3; f2:= f3; goto anew end;
      x0:= x4; f0:= f4
   end int;

   re:= e[1]; ae:= e[2] × 180 / abs(b - a); e[3]:= 0;
   hmin:= abs(b - a) × re; x:= x0:= a; f0:= fx; x:= x2:= b;
   f2:= fx; x:= x1:= (x0 + x2) × .5; f1:= fx; sum:= 0; int;
   qad:= sum / 180
end qad;
```

```
real procedure int(x, a, b, fx, eps, tol); value a, b, eps, tol;
real x, a, b, fx, eps, tol;
begin real sum, fa, fb, hmin;
   real array Tnieuw[0:0];

   boolean procedure s(n, c, d, T); integer n; real c, d; real array T;
   begin T[0]:= if c = a then fa else fb; s:= true
   end s;

   procedure orde opvoeren(n, k, a, b, eps, Toud, p);
   value n, k, a, b, eps; integer n, k; real a, b, eps;
   real array Toud; procedure p;
   begin integer i, t;
      real h, d1, d2, u, g;
      real array Fx[1:k - 1], Tnieuw[0:n];

      real procedure vul(i, j); value i, j; integer i, j;
      begin integer m;
         if i = j then
         begin x:= a + i × h; Fx[i]:= vul:= fx end
         else
         begin m:= (i + j) : 2;
            Fx[m]:= vul:= (vul(i, m - 1) + vul(m + 1, j)) × 0.5
         end
      end vul;

      boolean procedure schema(n, c, d, T); value n, c, d; integer n;
      real c, d; real array T;
      begin integer i, t;
         real u, v;
         schema:= true; if n > 0 then
         begin if p(n - 1, c, d, T) then
            begin real array Test[0:n - 1];
               t:= 3;
               u:= (T[0] + Fx[(c + d - 2 × a) / (2 × h)]) × 0.5;
               for i:= 0 step 1 until n - 1 do
               begin v:= Test[i]:= T[i]; T[i]:= u;
                  u:= u + (u - v) / t;
                  if if i < 1 then false else abs((T[i] - T[i -
                  1]) / (Test[i] - Test[i - 1])) > 16 / (t + 1)
                  then
                  begin schema:= false; goto end end;
                  t:= 4 × t + 3
               end;
               T[n]:= u
            end
            else schema:= false
         end
         else
         begin t:= (c - a) / h; i:= t : 2 × 2 + 1; if n = 0 then
            begin if t < i then p( - 1, c, d, T) else p( - 1, d,
               c, T); T[0]:= (T[0] + Fx[i]) × 0.5
            end
            else if t < i then p( - 1, c, d, T) else T[0]:= Fx[i]
         end;
      end:
      end schema;
```

```
        procedure helft(n, k, a, b, eps); value n, k, a, b, eps;
        integer n, k; real a, b, eps;
        begin real m;
            if schema(n - 1, a, b, Tnieuw) then
            begin if if n < 2 then true else abs(Tnieuw[n - 1] -
                Tnieuw[n - 2]) > eps then orde opvoeren(n, k, a, b,
                eps, Tnieuw, schema) else sum:= sum + (b - a) ×
                Tnieuw[n - 1]
            end
            else
            begin m:= (a + b) × 0.5; k:= k : 2; eps:= eps × 0.5;
                helft(n - 1, k, a, m, eps);
                helft(n - 1, k, m, b, eps)
            end
        end helft;

        h:= (b - a) / k; if h < hmin then goto end;
        u:= d1:= (Toud[0] + vul(1, k - 1)) × 0.5; t:= 3;
        for i:= 0 step 1 until n - 1 do
        begin d2:= (u - Toud[i]) / t; Tnieuw[i]:= u;
            if if i < 1 then false else abs((Tnieuw[i] - Tnieuw[i -
            1]) / (Toud[i] - Toud[i - 1])) > 16 / (t + 1) then goto
            opsplitsen; u:= u + d2; d1:= d2; t:= 4 × t + 3
        end;
        if abs(d2) < eps + (b - a) × tol then sum:= sum + (b - a) ×
        u else
        begin Tnieuw[n]:= u;
            orde opvoeren(n + 1, 2 × k, a, b, eps, Tnieuw, schema)
        end;
        goto end;
    opsplitsen: if abs(d1) < eps + (b - a) × tol then sum:= sum
        + (b - a) × u else
        begin g:= (a + b) × 0.5; eps:= eps × 0.5;
            helft(n, k, a, g, eps); helft(n, k, g, b, eps)
        end;
    end:
    end orde opvoeren;

    x:= a; fa:= fx; x:= b; fb:= fx; Tnieuw[0]:= (fa + fb) × 0.5;
    if tol < 10 - 12 then tol:= 10 - 12; hmin:= (b - a) × tol;
    sum:= 0; orde opvoeren(1, 2, a, b, eps / (b - a), Tnieuw, s);
    int:= sum
end int;
```

```
real procedure integral(x, a, b, fx, ae, re, max, full up);
value a, b, ae, re, max; real x, a, b, fx, ae, re; integer max;
label full up;
begin integer r, su, l, i, j, lpo, lpi, rpi, rpo, lpa, s, n;
   real h, hh, hmin, tr, sum, t0, t1, s0, s1, d, c, ba, lva, rva,
   mva;
   boolean cl, left, fr;
   array fa[0:max], t[0:119];

   procedure rom;
   begin integer llp;
      real llv, ltr, lt0, lt1;
      boolean lcl;
      if r = - 1 then
      begin lva:= rva; x:= x + h; mva:= fx; rpi:= rpi + s;
         rva:= fa[rpi]; lpa:= lpi; fa[lpi]:= lva; lpi:= lpi + s;
         fa[lpi]:= mva; lpi:= lpi + s; n:= n + 2;
         if (lpi - rpi) × s ≥ 0 then goto full up;
         t1:= (lva + rva) × .5; t0:= (t1 + mva) × .5;
         c:= (t0 - t1) / 3; t0:= t0 + c;
         if abs(c × ba) < abs(t0 × re) + ae then
         begin tr:= abs(h) × t0; cl:= true; n:= n - 2; b:= n × hh;
            if abs(b) > hmin then
            begin fa[lpi - s]:= a + b; fa[lpi]:= n;
               lpo:= lpi:= lpi + s
            end
            else lpi:= lpo; n:= 0; a:= x + hh
         end
         else
         begin tr:= 0; cl:= false end
      end
      else
      begin j:= j - r; r:= r - 1; l:= l / 2; left:= true; rom;
      sr: llp:= lpa; llv:= lva; ltr:= tr; lcl:= cl; lt1:= t1;
         lt0:= t0; left:= false; rom; l:= l + 1; r:= r + 1;
         j:= j + r; lva:= llv; lpa:= llp; s0:= (lva + rva) × .5;
         t[j + r]:= if left then (s0 × .5) else (t[j + r] + s0 ×
         .5); d:= 1;
         for i:= j + r - 1 step - 1 until j do
         begin d:= d × 4; s1:= s0; s0:= t[i - r];
            c:= (s0 - s1) / (d - 1); s0:= s0 + c;
            t[i]:= if left then (s0 × .5) else (t[i] + s0 × .5)
         end;
         d:= d × 4; s1:= s0; s0:= (lt1 + t1) × .5;
         c:= (s0 - s1) / (d - 1); s0:= s0 + c; d:= d × 4; t1:= s0;
         s0:= (lt0 + t0) × .5; c:= (s0 - t1) / (d - 1);
         t0:= s0 + c; if lcl ∧ cl then
         begin tr:= tr + ltr; cl:= true end
         else if abs(c × ba) < abs(t0 × re) + ae then
         begin tr:= abs(h) × t0 × l; cl:= true;
         anew: n:= (lpa - lpo) × s; if n ≥ 0 then
            begin b:= n × hh; if abs(b) > hmin then
               begin lpi:= lpa + s; fa[lpi]:= a + b; lpi:= lpi + s;
                  fa[lpi]:= n; lpo:= lpi:= lpi + s
               end
               else lpi:= lpo; n:= 0; a:= x + hh
```

```
            end
            else
            begin n:= fa[lpo - s]; b:= fa[lpo - s - s];
               a:= b - n × hh; lpo:= lpo - (n + 3) × s; goto anew
            end
         end
         else
         begin tr:= tr + ltr; cl:= false end
      end;
      if r = su then
      begin su:= su + 1; if abs(rpo - rpi) ≥ 1 then goto sr end
   end;

   procedure suprom; if rpi ≠ rpo then
   begin r:= su:= - 1; l:= 1; j:= 0; left:= true; rom;
      sum:= sum + tr; suprom
   end
   else if n ≠ 0 then
   begin fa[lpi]:= rva; lpi:= lpi + s; b:= n × hh;
      if abs(b) > hmin then
      begin fa[lpi]:= a + b; lpi:= lpi + s; fa[lpi]:= n;
         lpo:= lpi:= lpi + s
      end
      else lpi:= lpo
   end;

   procedure subinterval;
anew: if rpo = 0 ∨ rpo = max then
   begin if ¬fr then
      begin fr:= true; h:= - hh; hh:= - .5 × hh; s:= - s;
         lpi:= rpi; rpi:= lpo; lpo:= rpo; rpo:= rpi; goto anew
      end
   end
   else
   begin fr:= false; rpi:= rpi + s; n:= fa[rpi]; rpi:= rpi + s;
      a:= fa[rpi]; rpi:= rpi + s; rva:= fa[rpi];
      rpo:= rpi + n × s; n:= 0; x:= a - hh; suprom; goto anew
   end;

   ba:= b - a; if b < a then
   begin h:= b; b:= a; a:= h end;
   h:= abs(ba); hh:= h × .5; hmin:= abs(ba × re); x:= b;
   fa[max]:= fx; x:= a; fa[max - 1]:= fx; fa[max - 2]:= a;
   fa[max - 3]:= 1; rpo:= rpi:= max - 4; lpo:= lpi:= 0; s:= 1;
   sum:= 0; subinterval; integral:= sign(ba) × sum
end;
```

```
real procedure qadrat(x, a, b, fx, e); value a, b; real x, a, b, fx;
array e;
begin real f0, f2, f3, f5, f6, f7, f9, f14, v, w, hmin, hmax, re,
   ae;

   real procedure lint(x0, xn, f0, f2, f3, f5, f6, f7, f9, f14);
   real x0, xn, f0, f2, f3, f5, f6, f7, f9, f14;
   begin real h, xm, f1, f4, f8, f10, f11, f12, f13;
      xm:= (x0 + xn) / 2; h:= (xn - x0) / 32; x:= xm + 4 × h;
      f8:= fx; x:= xn - 4 × h; f11:= fx; x:= xn - 2 × h; f12:= fx;
      v:= 0.3305801781992 × f7 + 0.1734851157073 × (f6 + f8) +
      0.3211054265600 × (f5 + f9) + 0.1350077083410 × (f3 + f11)
      + 0.1657145142282 × (f2 + f12) + 0.3939714606381₁₀ - 1 × (f0
      + f14); x:= x0 + h; f1:= fx; x:= xn - h; f13:= fx;
      w:= 0.2606524413236 × f7 + 0.2390632833514 × (f6 + f8) +
      0.2630626354775 × (f5 + f9) + 0.2186819313831 × (f3 + f11)
      + 0.2757897646643₁₀ - 1 × (f2 + f12) + 0.1055750100538 × (f1
      + f13) + 0.1571194260595₁₀ - 1 × (f0 + f14);
      if abs(h) < hmin then e[3]:= e[3] + 1;
      if abs(v - w) < abs(w) × re + ae ∨ abs(h) < hmin then
      begin lint:= h × w; goto out end;
      x:= x0 + 6 × h; f4:= fx; x:= xn - 6 × h; f10:= fx;
      v:= 0.2456734301503 × f7 + 0.2557862582869 × (f6 + f8) +
      0.2285260636904 × (f5 + f9) + 0.5005571315559₁₀ - 1 × (f4 +
      f10) + 0.1779464877368 × (f3 + f11) + 0.5840145990321₁₀ - 1
      × (f2 + f12) + 0.8748309428713₁₀ - 1 × (f1 + f13) +
      0.1896420786481₁₀ - 1 × (f0 + f14);
      lint:= if abs(v - w) < abs(v) × re + ae then h × v else
      lint(x0, xm, f0, f1, f2, f3, f4, f5, f6, f7) + rint(xm,
      xn, f7, f8, f9, f10, f11, f12, f13, f14);
   out:
   end lint;


   real procedure rint(x0, xn, f0, f5, f7, f8, f9, f11, f12, f14);
   real x0, xn, f0, f5, f7, f8, f9, f11, f12, f14;
   begin real h, xm, f1, f2, f3, f4, f6, f10, f13;
      xm:= (x0 + xn) / 2; h:= (xn - x0) / 32; x:= x0 + 2 × h;
      f2:= fx; x:= x0 + 4 × h; f3:= fx; x:= xm - 4 × h; f6:= fx;
      v:= 0.3305801781992 × f7 + 0.1734851157073 × (f6 + f8) +
      0.3211054265600 × (f5 + f9) + 0.1350077083410 × (f3 + f11)
      + 0.1657145142282 × (f2 + f12) + 0.3939714606638₁₀ - 1 × (f0
      + f14); x:= x0 + h; f1:= fx; x:= xn - h; f13:= fx;
      w:= 0.2606524413236 × f7 + 0.2390632833514 × (f6 + f8) +
      0.2630626354775 × (f5 + f9) + 0.2186819313831 × (f3 + f11)
      + 0.2757897646643₁₀ - 1 × (f2 + f12) + 0.1055750100538 × (f1
      + f13) + 0.1571194260595₁₀ - 1 × (f0 + f14);
      if abs(h) < hmin then e[3]:= e[3] + 1;
      if abs(v - w) < abs(w) × re + ae ∨ abs(h) < hmin then
      begin rint:= h × w; goto out end;
      x:= x0 + 6 × h; f4:= fx; x:= xn - 6 × h; f10:= fx;
      v:= 0.2456734301503 × f7 + 0.2557862582869 × (f6 + f8) +
      0.2285260636904 × (f5 + f9) + 0.5005571315559 × (f4 + f10)
      + 0.1779464877368 × (f3 + f11) + 0.5840145990321₁₀ - 1 × (f2
      + f12) + 0.8748309428713₁₀ - 1 × (f1 + f13) +
      0.1896420786481₁₀ - 1 × (f0 + f14);
```

```
        rint:= if abs(v - w) < abs(v) × re + ae then h × v else
        lint(x0, xm, f0, f1, f2, f3, f4, f5, f6, f7) + rint(xm,
        xn, f7, f8, f9, f10, f11, f12, f13, f14);
    out:
    end rint;

    re:= e[1]; ae:= 2 × e[2] / abs(b - a); e[3]:= 0;
    hmin:= abs(b - a) × re; hmax:= (b - a) / 16; x:= a; f0:= fx;
    x:= a + hmax; f2:= fx; x:= a + 2 × hmax; f3:= fx;
    x:= a + 4 × hmax; f5:= fx; x:= a + 6 × hmax; f6:= fx;
    x:= a + 8 × hmax; f7:= fx; x:= b - 4 × hmax; f9:= fx; x:= b;
    f14:= fx;
    qadrat:= lint(a, b, f0, f2, f3, f5, f6, f7, f9, f14) × 16
end qadrat;
```

```
real procedure trapex(x, fx, a, b, e); value a, b; real x, fx, a, b;
array e;
begin integer nn, nn1, nn2, i, orde, m;
   real f1, f2, f2a, f3, k1, k2, h0, h, to, tr, tn, ae, re;
   array t[0:e[3]];

   procedure extr(m); value m; integer m;
   begin integer i;
      real u, v, tu, tv, d, r1, r2, r3;
      v:= 0; u:= t[0]; tr:= t[0]:= tn; r1:= 2 / 3; r2:= .75;
      r3:= 1;
      for i:= 1 step 1 until m - 1 do
      begin if i : 2 × 2 = i then d:= r3:= r3 × 2 else if m : 2 ×
         2 = m then d:= r2:= r2 × 2 else d:= r1:= r1 × 2;
         d:= d × d; tv:= tr - v; tu:= tr - u;
         if tv ≠ 0 then tr:= tr + tu / (d × (1 - tu / tv) - 1);
         v:= u; u:= t[i]; t[i]:= tr
      end;
      if m : 2 × 2 = m then d:= k2:= k2 × 2 else d:= k1:= k1 × 2;
      d:= d × d; tv:= tr - v; tu:= tr - u;
      if tv ≠ 0 then tr:= tr + tu / (d × (1 - tu / tv) - 1);
      t[m]:= tr
   end extr;

   re:= e[1]; ae:= e[2]; orde:= e[3]; k1:= 1; k2:= 1.5; nn2:= 2;
   nn1:= 3; h0:= b - a; x:= a; f1:= fx; x:= b; f1:= (f1 + fx) / 2;
   t[0]:= f1 × h0; x:= a + h0 / 2; f2a:= f2:= fx;
   tn:= (f1 + f2) × h0 / 2; extr(1); to:= tr;
   for m:= 2 step 1 until orde do
   begin if m = 2 then
      begin x:= a + h0 / 3; f3:= fx; x:= b - h0 / 3; f3:= f3 + fx;
         tn:= (f1 + f3) × h0 / 3
      end
      else
      begin if nn2 < nn1 then
         begin nn:= nn2:= nn2 × 2; h:= h0 / nn end
         else
         begin nn:= nn1:= nn1 × 2; h:= h0 / nn end;
         if m = (m : 2) × 2 then
         begin for i:= 1 step 6 until nn, 5 step 6 until nn do
            begin x:= a + i × h; f3:= f3 + fx end;
            tn:= (f3 + f2a + f1) × h; f2a:= f2
         end
         else
         begin for i:= 1 step 2 until nn do
            begin x:= a + i × h; f2:= f2 + fx end;
            tn:= (f2 + f1) × h
         end
      end;
      extr(m);
      if abs(to - tr) ≤ ae + re × abs(tr) then goto end else
      to:= tr
   end;
   m:= 0;
end: trapex:= tr; e[3]:= m
end trapex;
```

3. Description of the procedures.

3.1. qad

The real procedure qad calculates the definite integral from a to b of the expression fx, depending on x.
Both a < b and a > b are allowed.
In array e[1:3] one must give the relative tolerance e[1] and the absolute tolerance e[2].
The procedure delivers the value of the required integral as value of the procedure identifier and in e[3] the number of steps satisfying

$$h < |b-a| * e[1],$$

where h is the length of the step.

The integral is calculated by means of Simpson's rule with Richardson correction. If the fourth difference is too large (and thus also the correction term), the total interval is split into two equal parts and the integration process is invoked recursively.
This is done in such a way that the total amount of Richardson corrections is slightly smaller than or equal to

$$e[1] * |\text{the integral from a to b of } |fx|\ | + e[1].$$

3.2. int

The real procedure int calculates the definite integral of fx from a to b, where fx is an expression, depending on x.
Both a < b and a > b are allowed.
One must give the relative tolerance tol and the absolute tolerance eps.
The procedure delivers the value of the required integral as value of the procedure identifier.
The integral is calculated by means of Romberg's method.
During the construction of the Romberg scheme splitting of the interval into two intervals of equal length is taken into consideration. In that case the integration process is invoked recursively.
If the correction term over some subinterval is sufficiently small, then the integration of that subinterval is finished.

3.3. integral.

The real procedure integral calculates the definite integral from a to b of the expression fx, depending on x.
Both a < b and a > b are allowed.
One must give the relative tolerance re and the absolute tolerance ae.

The integral is calculated by means of Romberg's method.
The Romberg scheme belonging to the whole interval is derived from the Romberg schemes belonging to its left and right halves, which in their turn are derived from Romberg schemes belonging to left and right halves and so on.
If the correction term over the whole interval is too large and the correction term over some subinterval is sufficiently small, then the integration of that subinterval is finished.
The same procedure is applied to the remaining subintervals.
Intervals are represented by a set of function values stored contiguously in the array fa [0:max].
The procedure delivers the value of the integral as value of the procedure identifier, unless the procedure fails. In that case the procedure body is left prematurely and the next statement of the program to be executed is the one labeled with the actual parameter belonging to full up.

3.4. qadrat.

The real procedure qadrat calculates the definite integral of fx from a to b, where fx is an expression, depending on x.
Both a < b and a > b are allowed.
In array e[1:3] one must give the relative tolerance e[1] and the absolute tolerance e[2].
The procedure delivers the value of the required integral as value of the procedure identifier and in e[3] the number of steps, satisfying

$$h < |b - a| * e[1],$$

where h is the length of the step.
By means of formulas of order 12 and 14, approximations of the integral, resp. I, and I_2, are calculated.
If

$$|I_1 - I_2| \geq |I_2| * e[1] + e[2],$$

then I_3, an approximation of the integral by means of a 16th order formula is calculated.
If furthermore

$$|I_2 - I_3| \geq |I_3| * e[1] + e[2],$$

then the concerned interval is split into two intervals of equal length and the integration process is invoked recursively.

3.5. trapex.

The real procedure trapex calculates the definite integral of fx from a to b, where fx is an expression depending on x.
Both a < b and b < a are allowed.
In array e[1:3] one must give the relative tolerance e[1], the absolute tolerance e[2], and the maximal number of subdivisions of the integration interval e[3].
The procedure delivers the value of the required integral as value of the procedure identifier and the number of subdivisions of the integration interval in e[3].
The method is as follows. For a series of integers n_k, where

$$n_0 = 1,\ n_1 = 2,\ n_2 = 3 \text{ and } n_k = 2n_{k-2} \text{ for } k = 3,\ldots, e[3],$$

the procedure calculates the trapezoidal approximation $T(h_k)$, where $h_k = (b-a)/n_k$ and improves the values $T(h_k)$ by extrapolation with rational functions.
The process is completed as soon as for some $k \geq 2$, the approximation T_k satisfies

$$|T_k - T_{k-1}| < e[1] * |T_k| + e[2],$$

and then T_k is delivered as value of the required integral and e[3]:= k.
If, however, the above condition is not satisfied for any $k \leq e[3]$, the procedure delivers $T_{e[3]}$ and e[3]:= 0.
The procedure is due to R. Bulirsch and J. Stoer (Num. Math., 1964, page 413 - 427), apart from non - essential modifications.

4. Test results

The procedures were tested on the following integrals

1. $\int_{.01}^{1.1} \frac{dx}{x^3}$ 2. $\int_{.01}^{1.1} \frac{dx}{x^4}$ 3. $\int_{.01}^{1.1} \frac{dx}{x^5}$

4. $\int_{0}^{1} \frac{dx}{1+x}$ 5. $\int_{0}^{1} \frac{dx}{1+x^4}$ 6. $\int_{1}^{10} \ln(x)dx$

7. $\int_{-1}^{1} \frac{dx}{x^2+10^{-2}}$ 8. $\int_{-1}^{1} \frac{dx}{x^2+10^{-3}}$ 9. $\int_{-1}^{1} \frac{dx}{x^2+10^{-4}}$

10. $\int_{-1}^{1} \frac{dx}{x^2+10^{-6}}$ 11. $\int_{-9}^{100} \frac{dx}{\sqrt{|x|}}$ 12. $\int_{0}^{1} \frac{dx}{1+5x^2}$

13. $\int_{0}^{1} \frac{dx}{1+10x^2}$ 14. $\int_{0}^{1} x^{\frac{1}{2}}\,dx$ 15. $\int_{0}^{1} x^{\frac{1}{5}}\,dx$

16. $\int_{0}^{1} x^{\frac{1}{10}}\,dx$ 17. $\int_{-1}^{1} |x+.5|^{\frac{1}{2}}\,dx$ 18. $\int_{0}^{1} \frac{dx}{1-.5x^2}$

19. $\int_{0}^{1} \frac{dx}{1-.98x^2}$ 20. $\int_{0}^{1} \frac{dx}{1-.998x^2}$ 21. $\int_{1/20}^{1/3} \frac{\sin(\frac{1}{x})}{x}\,dx$

22. $\int_{10^{-4}}^{7} \ln(x)\sin(x)dx$ 23. $\int_{0}^{1} (e^{-x}-e^{-10x})dx$

24. $\frac{2}{\sqrt{\pi}} \int_{0}^{1} (e^{-9x^2}+e^{-1024(x-.25)^2})dx$

25. $\int_{-1}^{2} \phi(x)dx, \quad \phi(x) = \begin{cases} e^{x} & \text{if } x \leq 0 \\ e^{1-x} & \text{if } x > 0 \end{cases}$

26. $\int_{-1}^{1.5} \theta(x)dx, \quad \theta(x) = \begin{cases} e^{10x} & \text{if } x \leq .5 \\ e^{10(1-x)} & \text{if } x > .5 \end{cases}$

For all integrals the tolerances are chosen such that

relative tolerance = absolute tolerance = $10^{-2}, 10^{-3}, \ldots, 10^{-8}$.

For each integral the results are shown in graphical form in figs. 1 - 26. The number of each figure refers to the number of the concerning integral. The meaning of the marks in the figures is as follows

legend:

+	qad
x	int
*	integral
Y	qadrat
□	trapex

In general for each procedure 7 marks, belonging to the given tolerances, are plotted. The exceptions to this rule are considered in chapter 5. The maximal number of subdivisions of trapex is 20; then the maximal number of functionevaluaties is 2049.

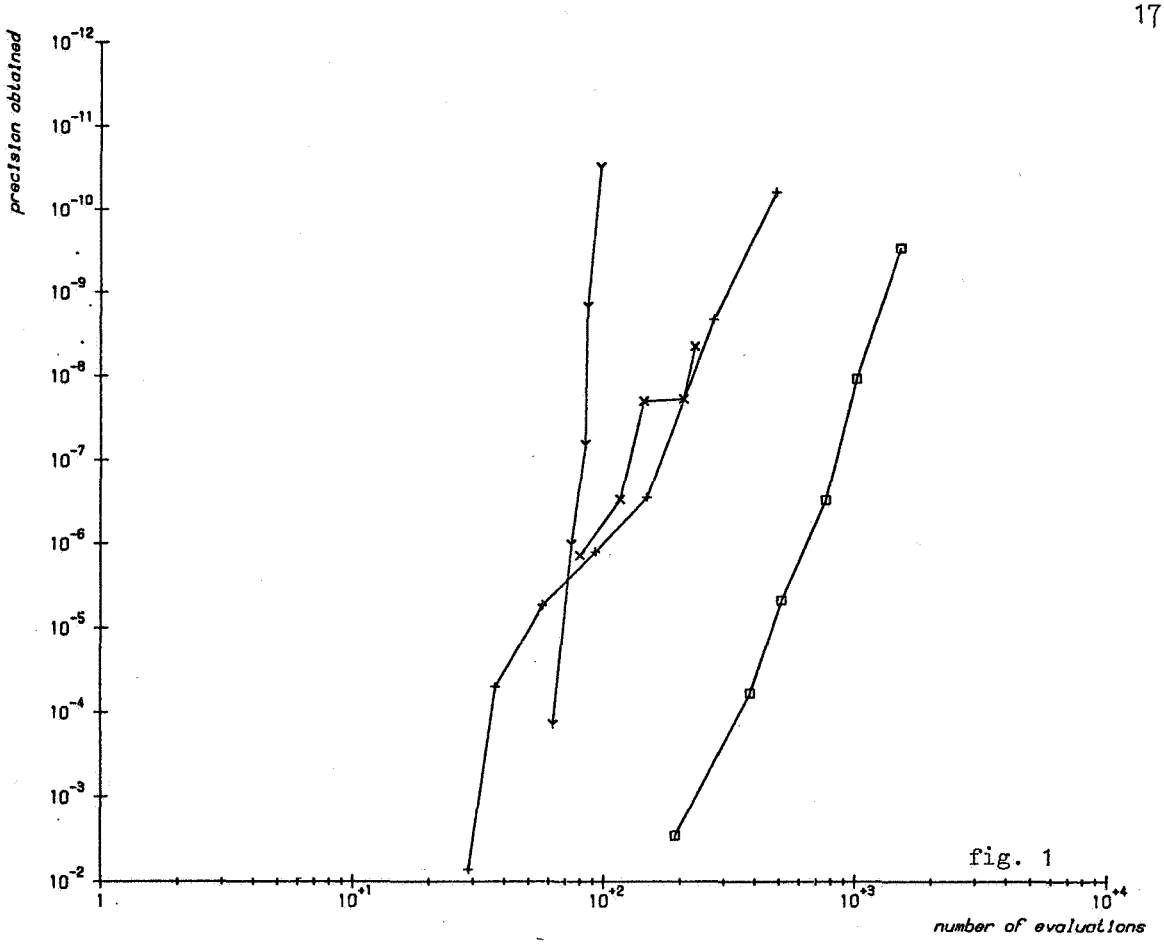
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 1

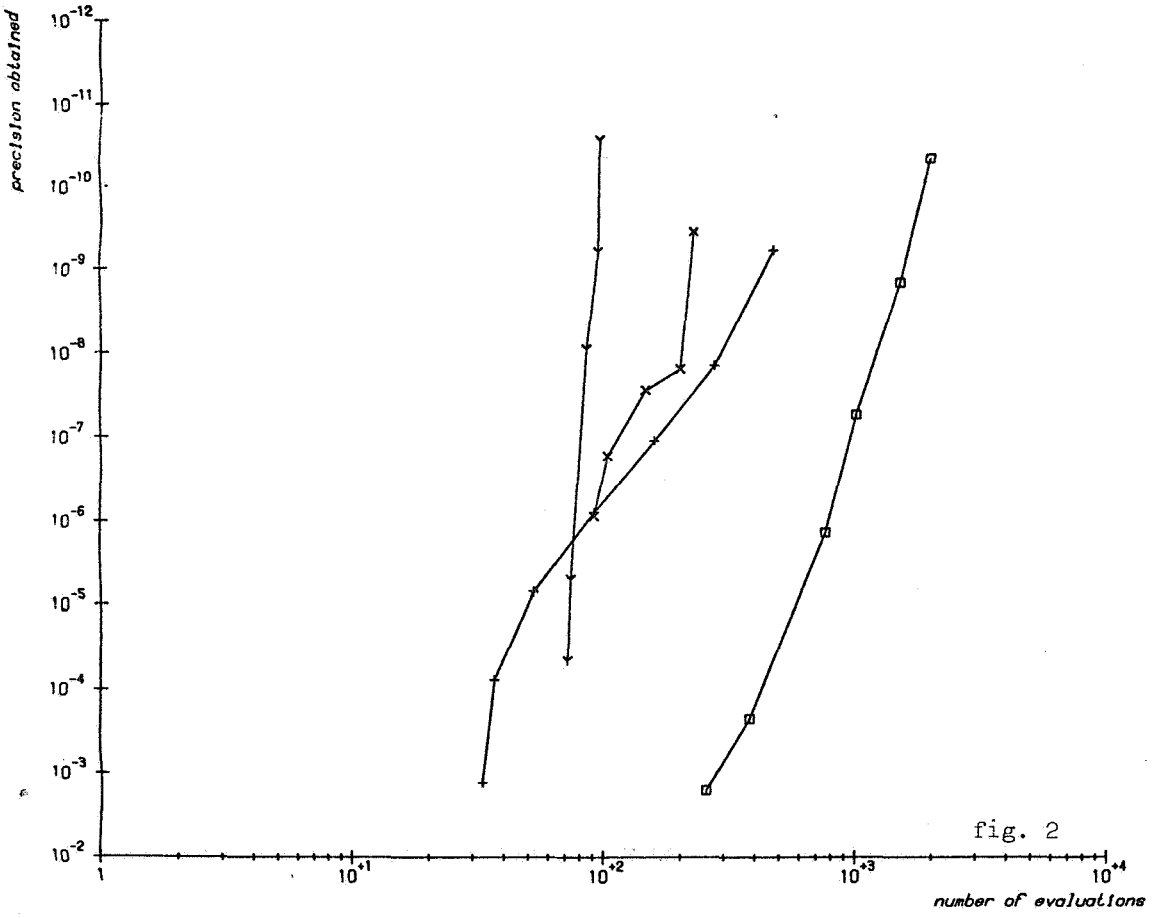
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 2

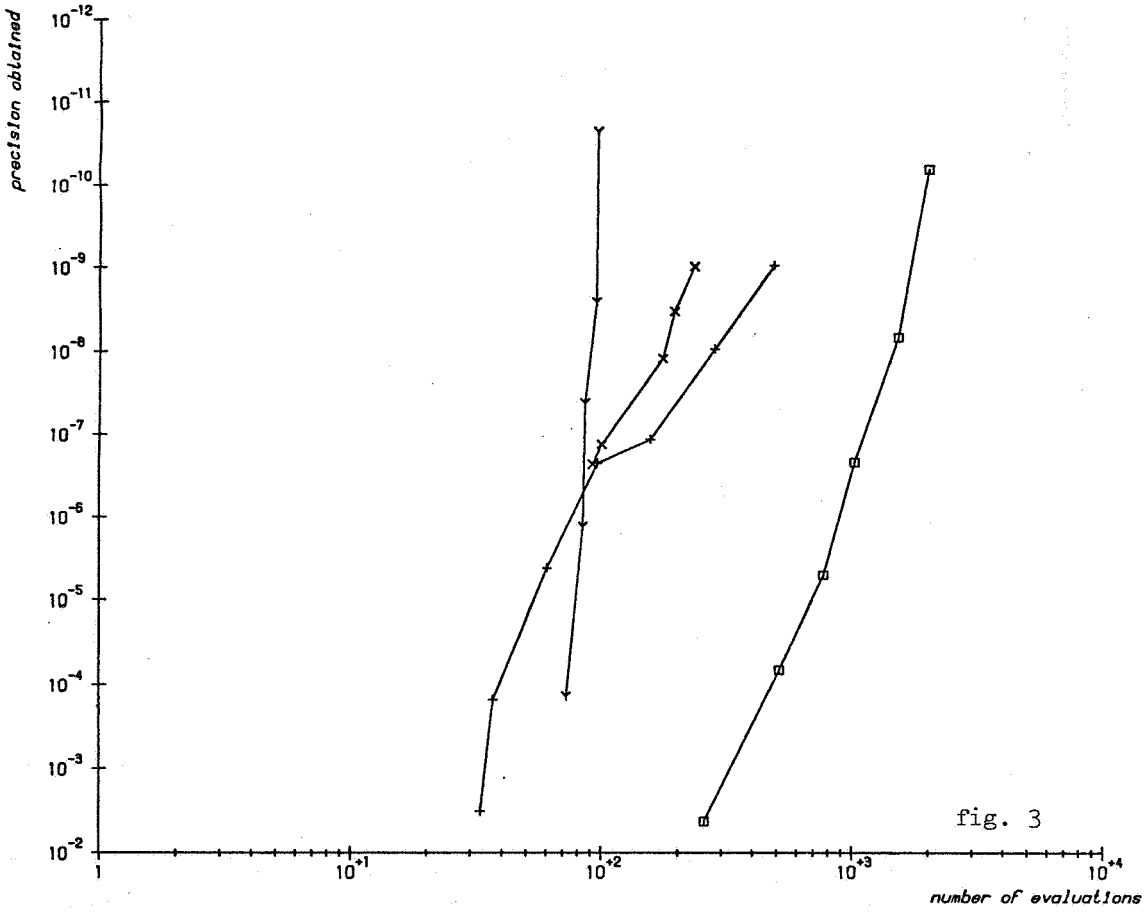
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 3

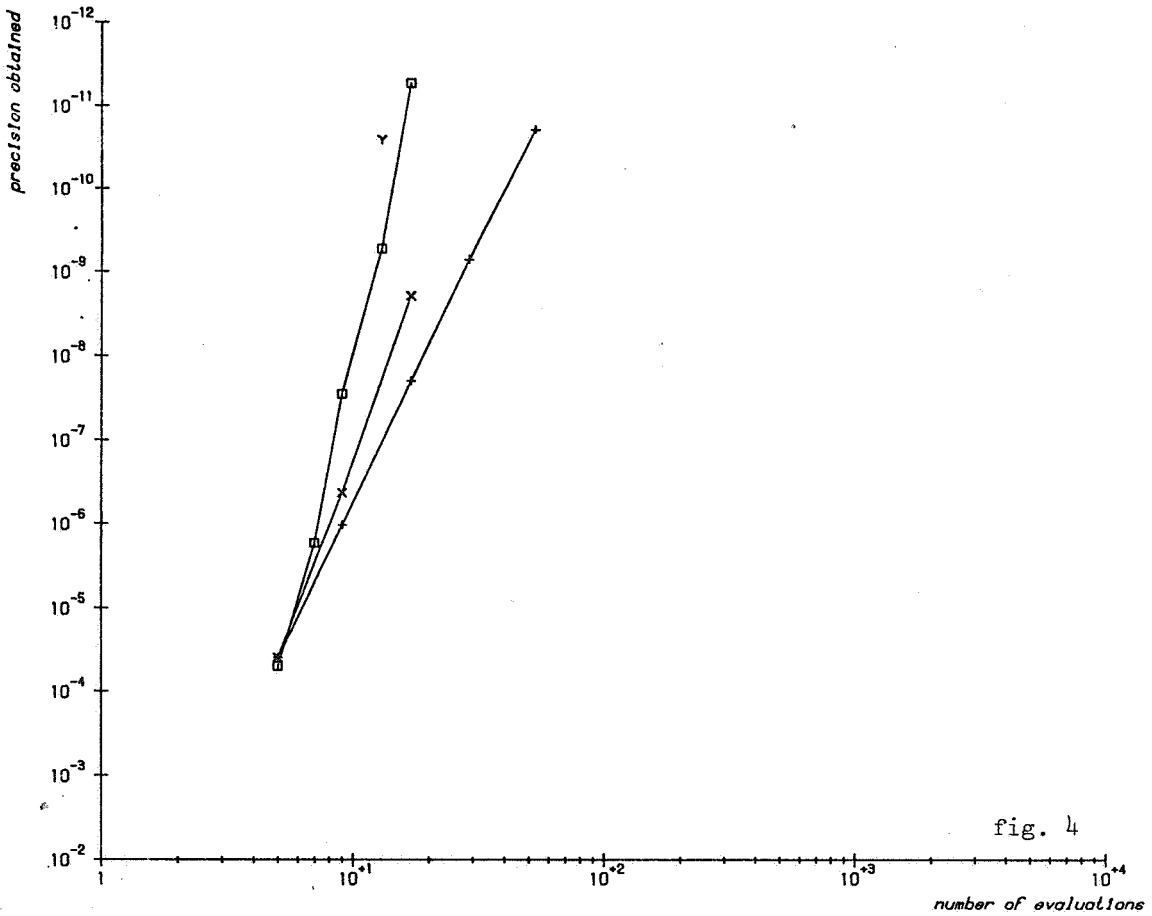
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 4

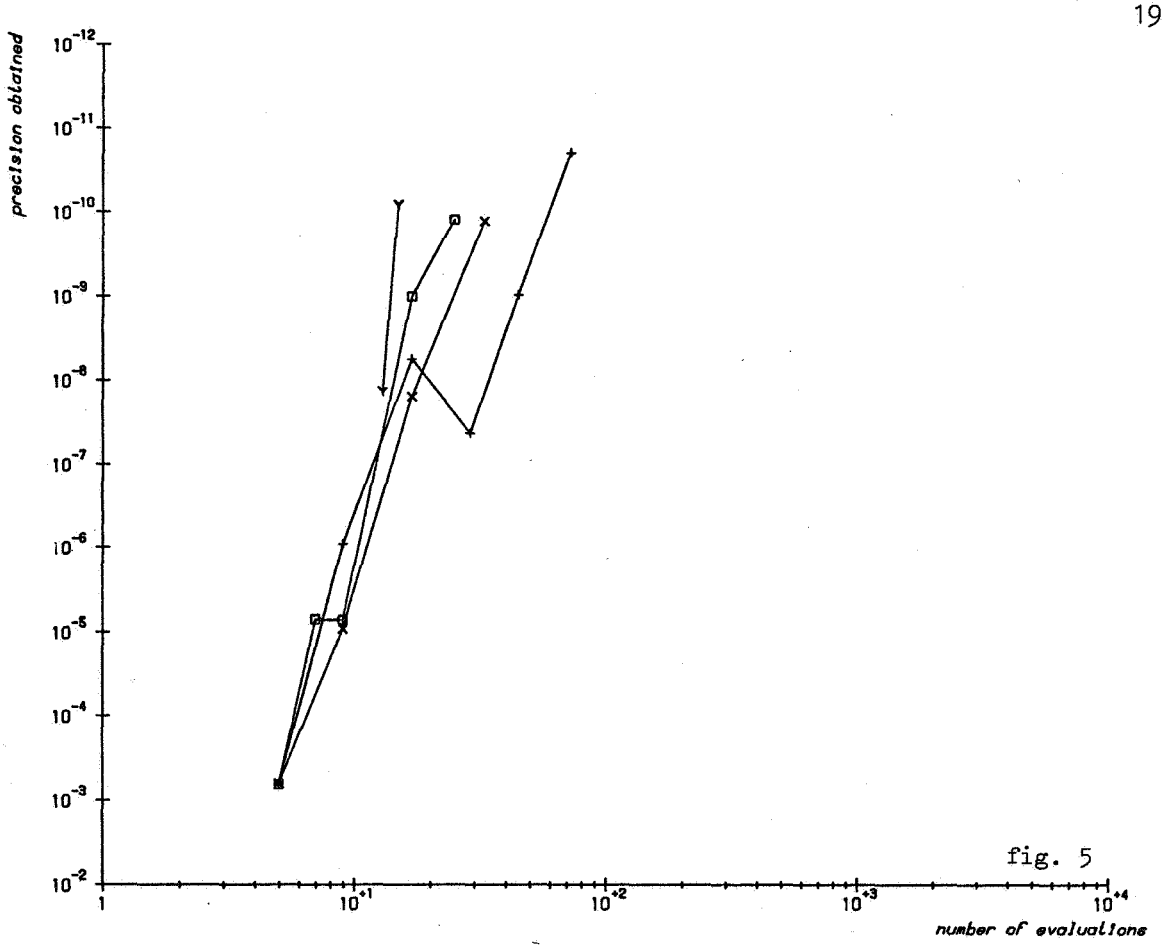
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 5

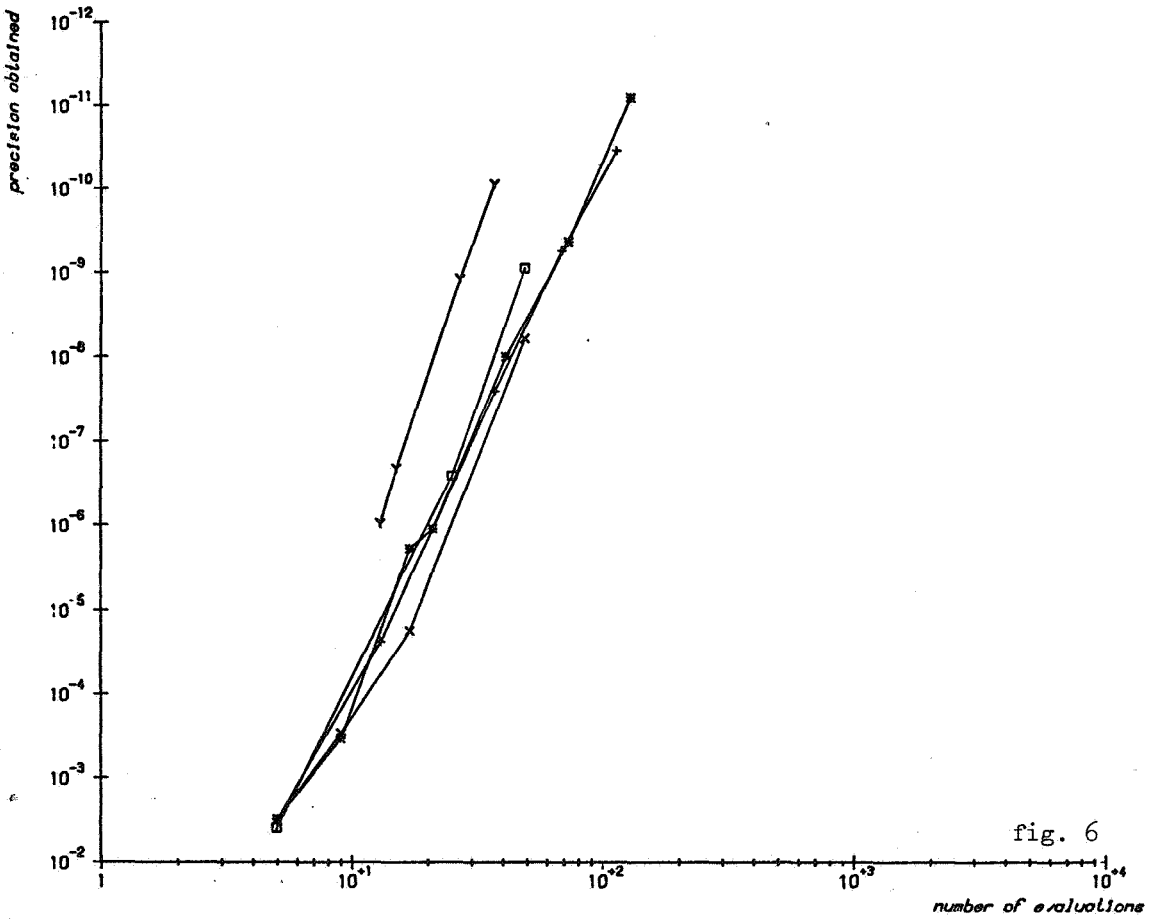
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 6

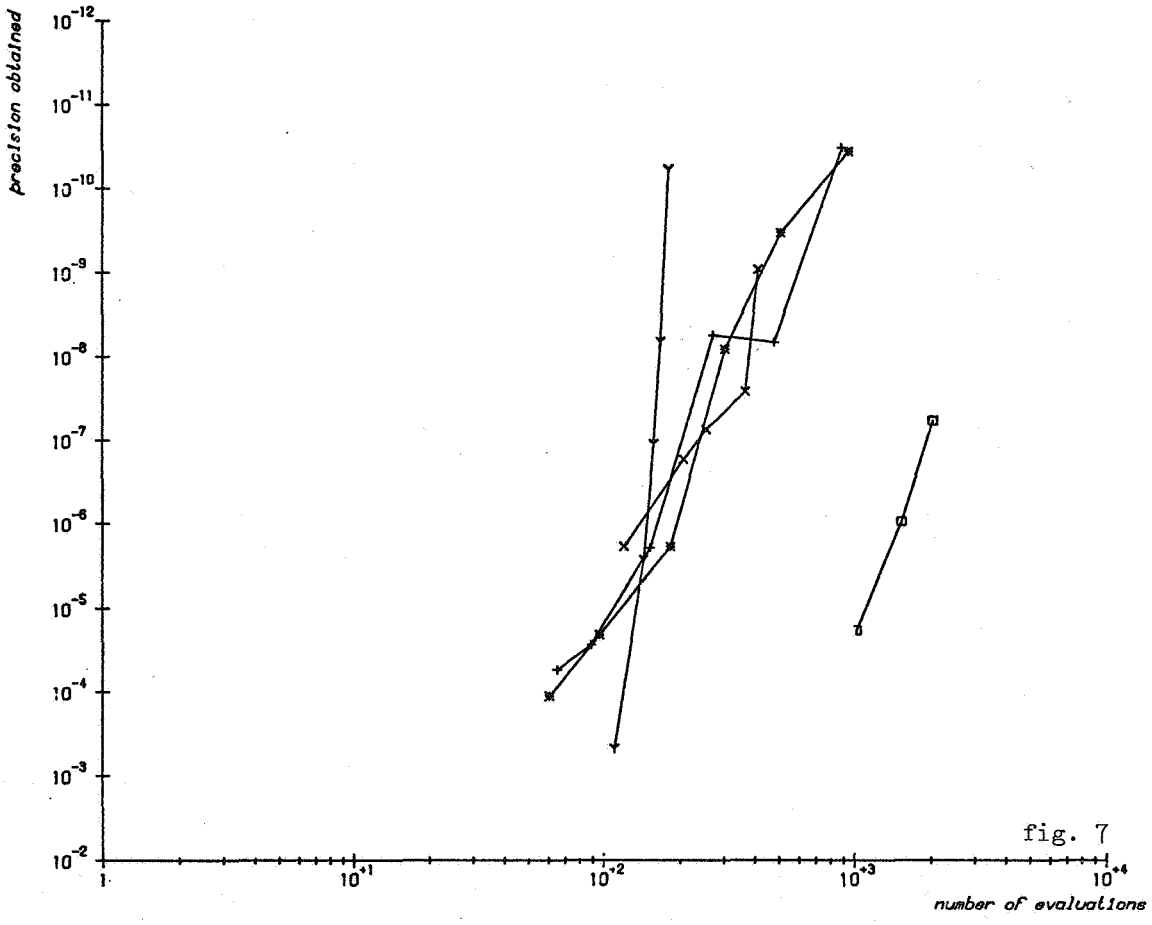
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 7

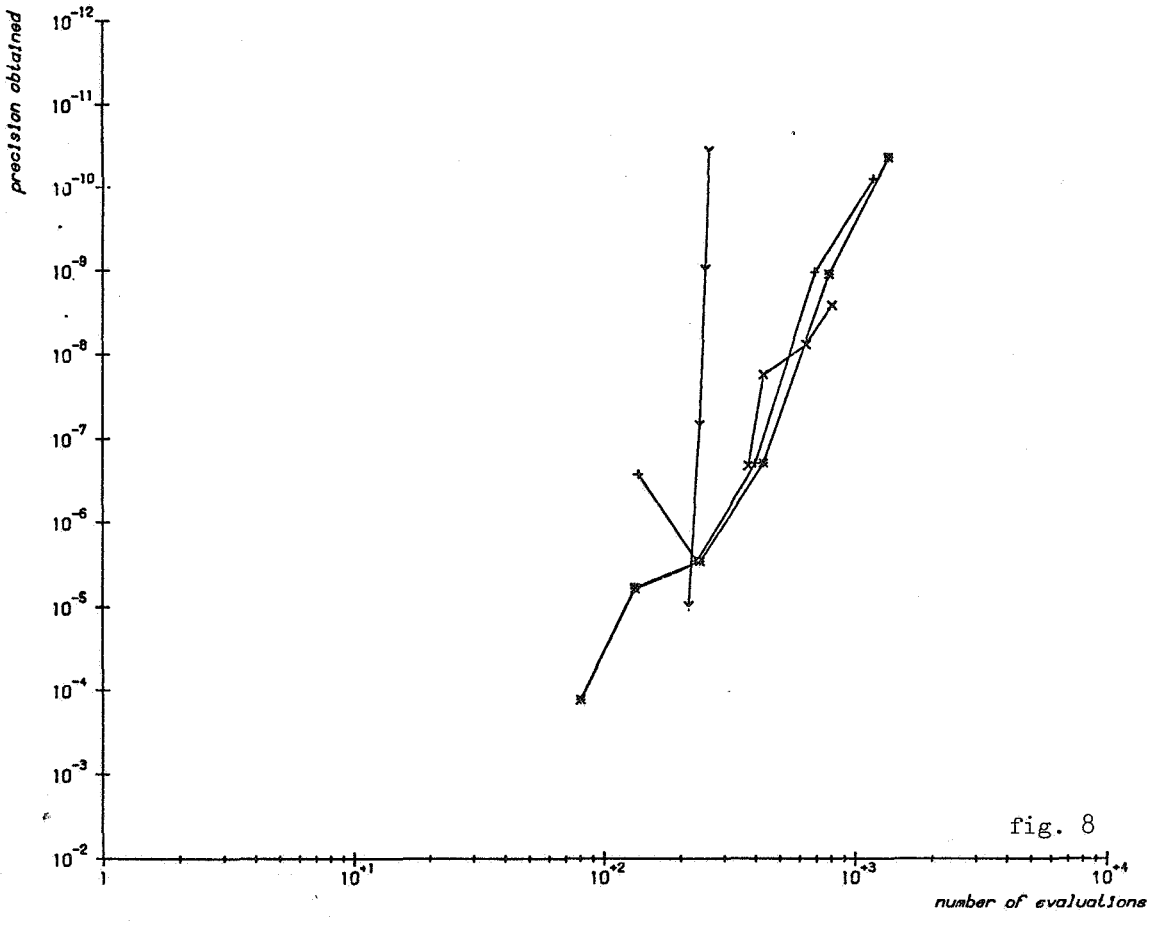
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 8

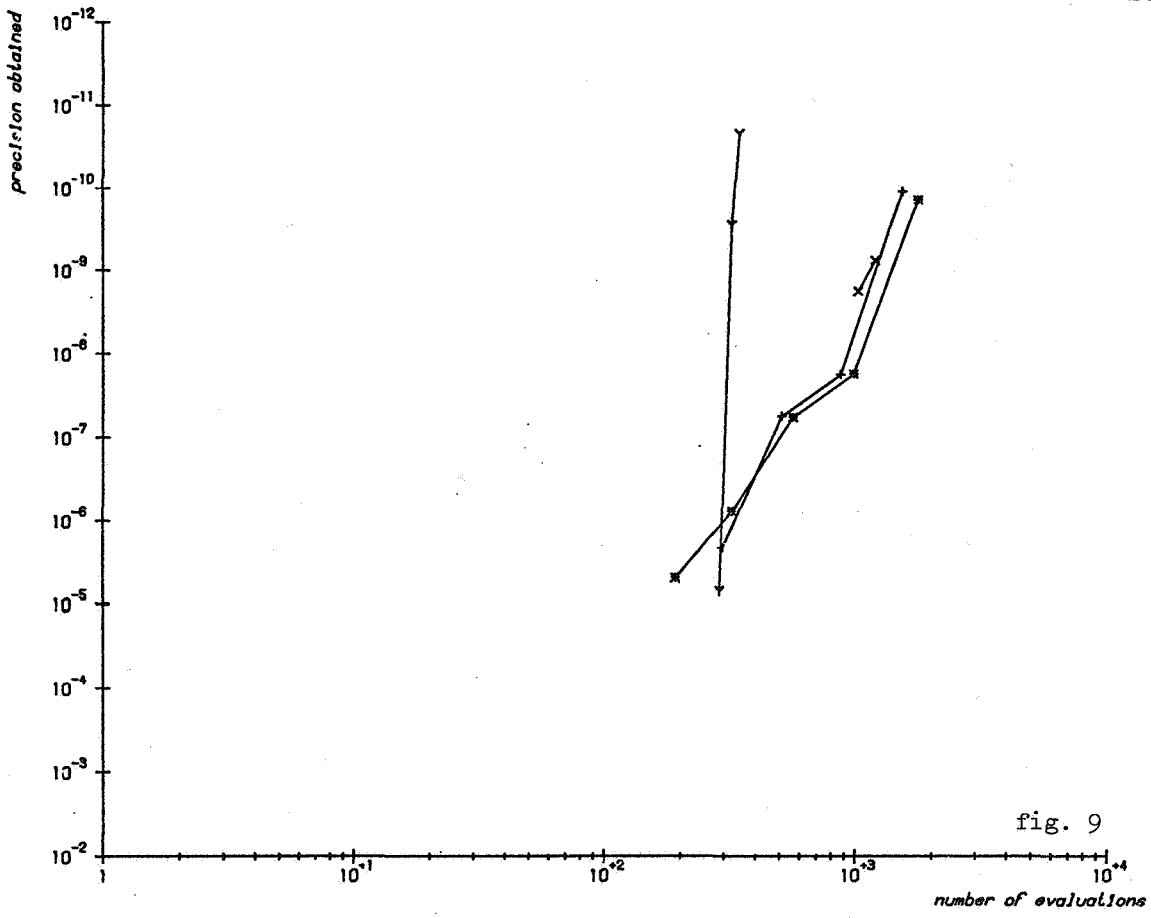
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 9

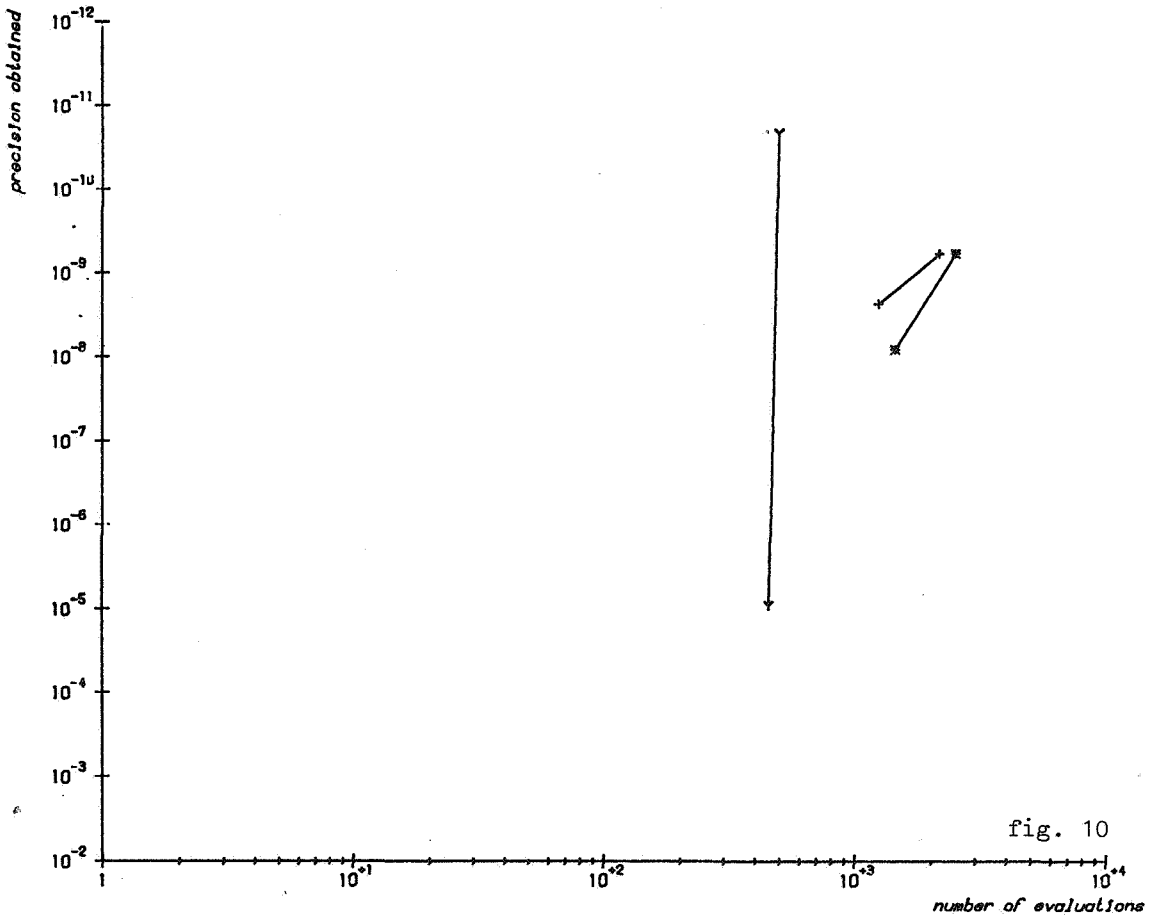
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 10

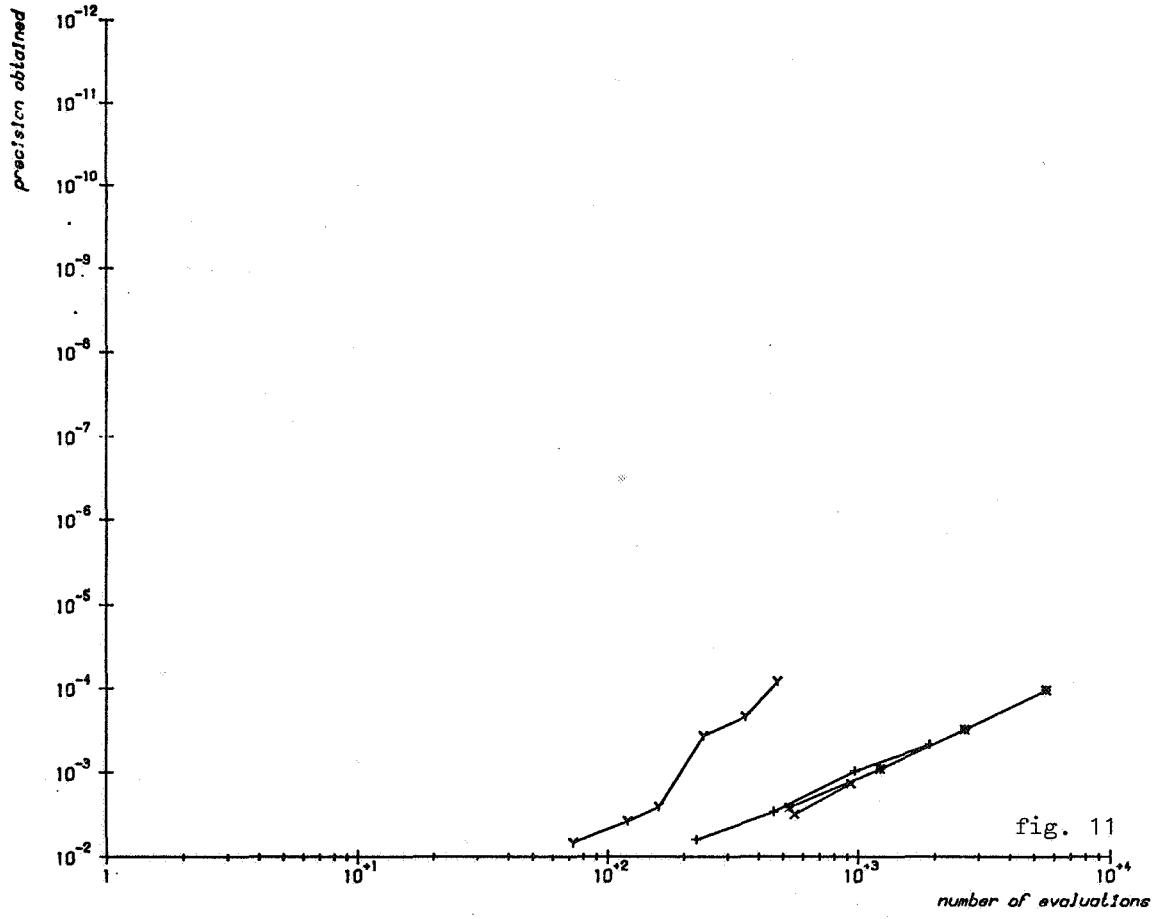
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 11

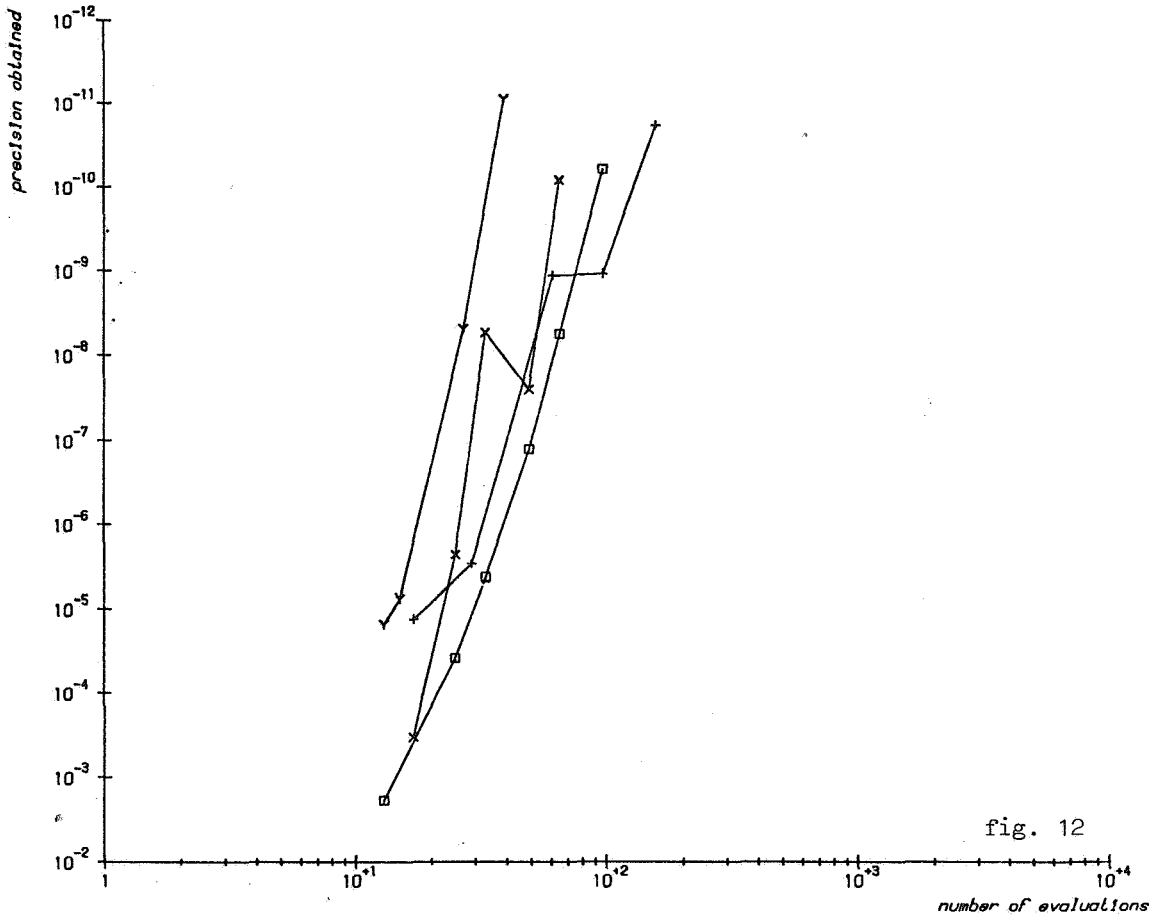
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 12

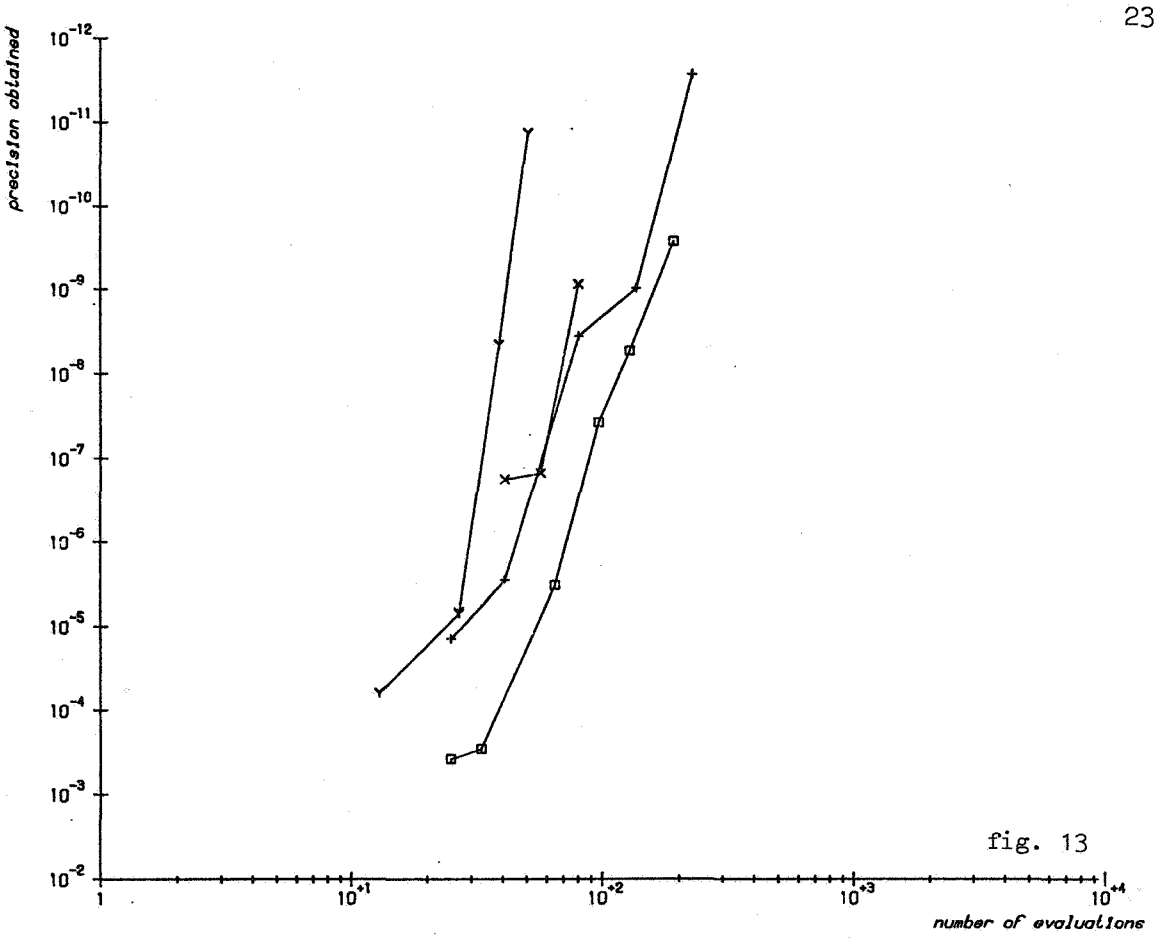
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 13

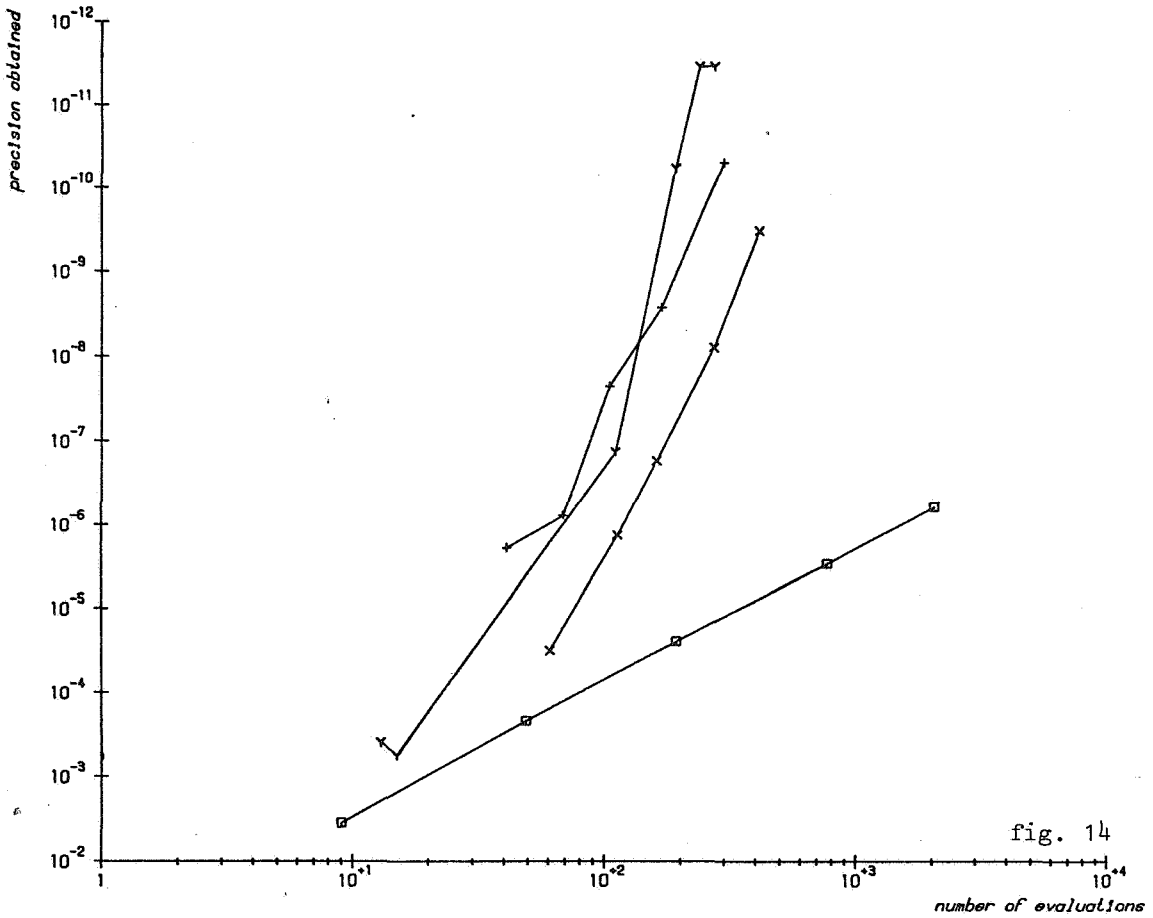
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 14

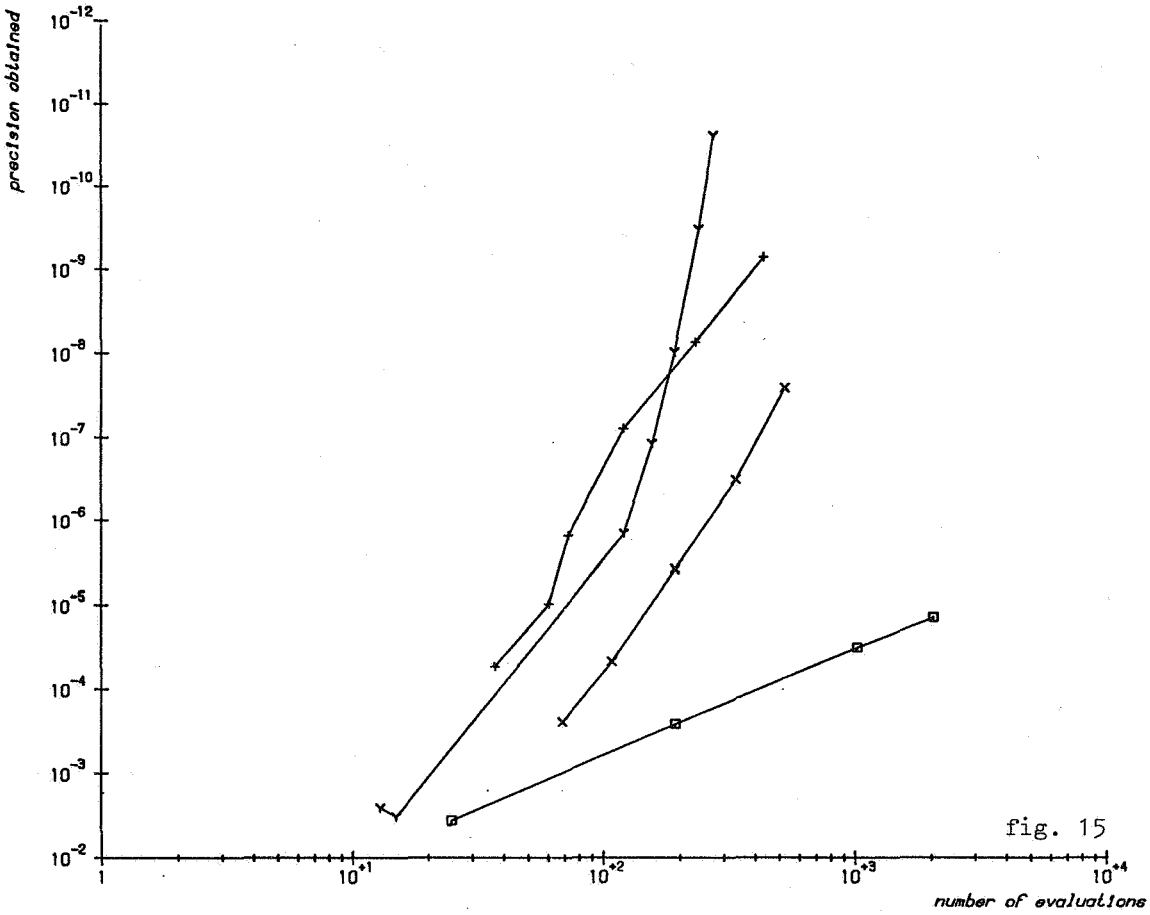
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 15

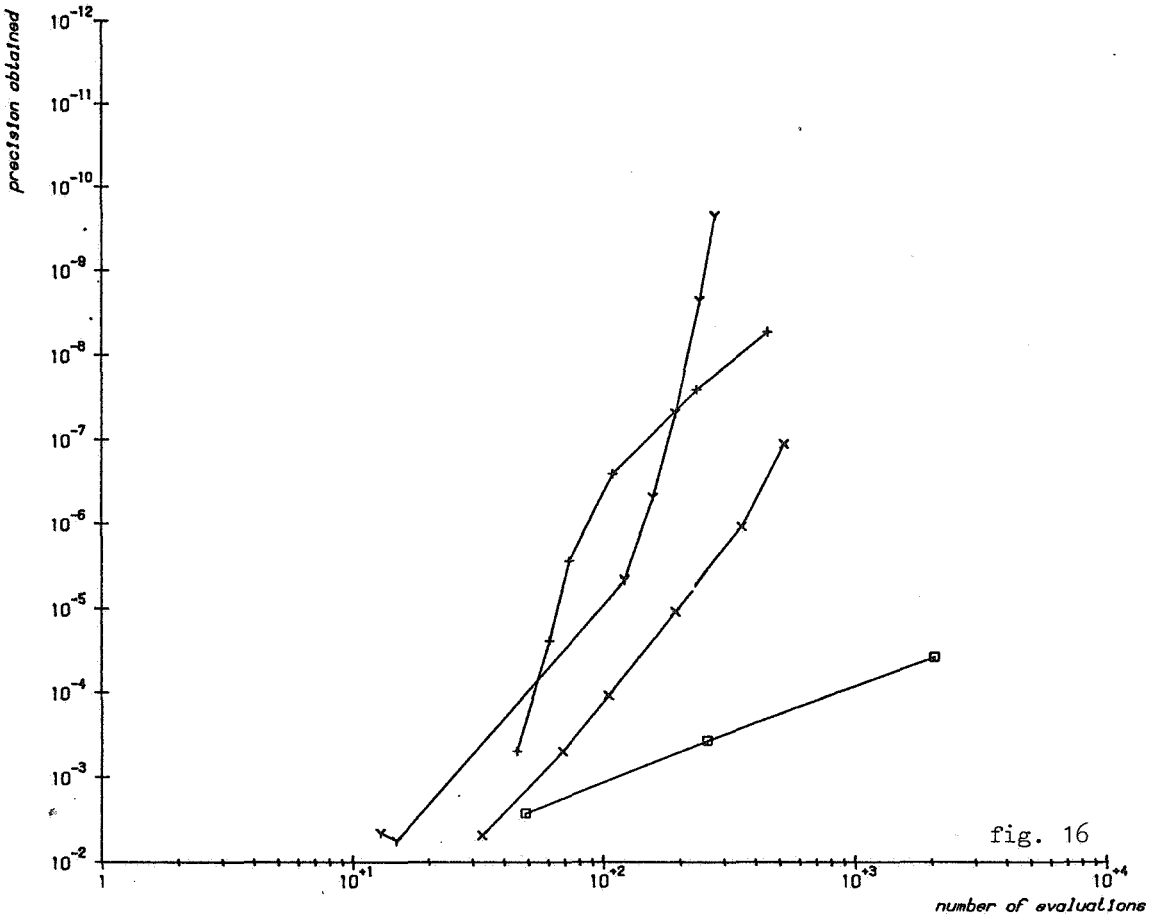
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 16

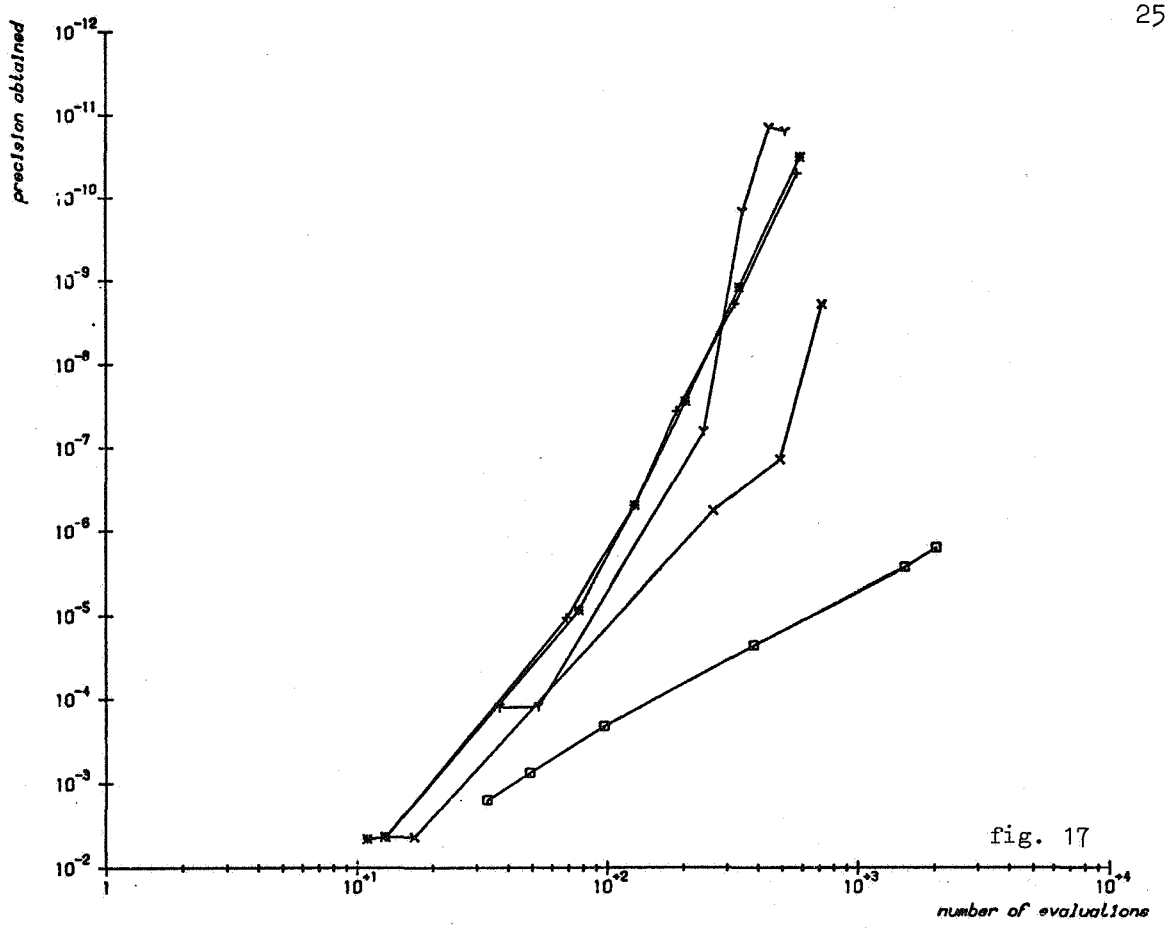
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 17

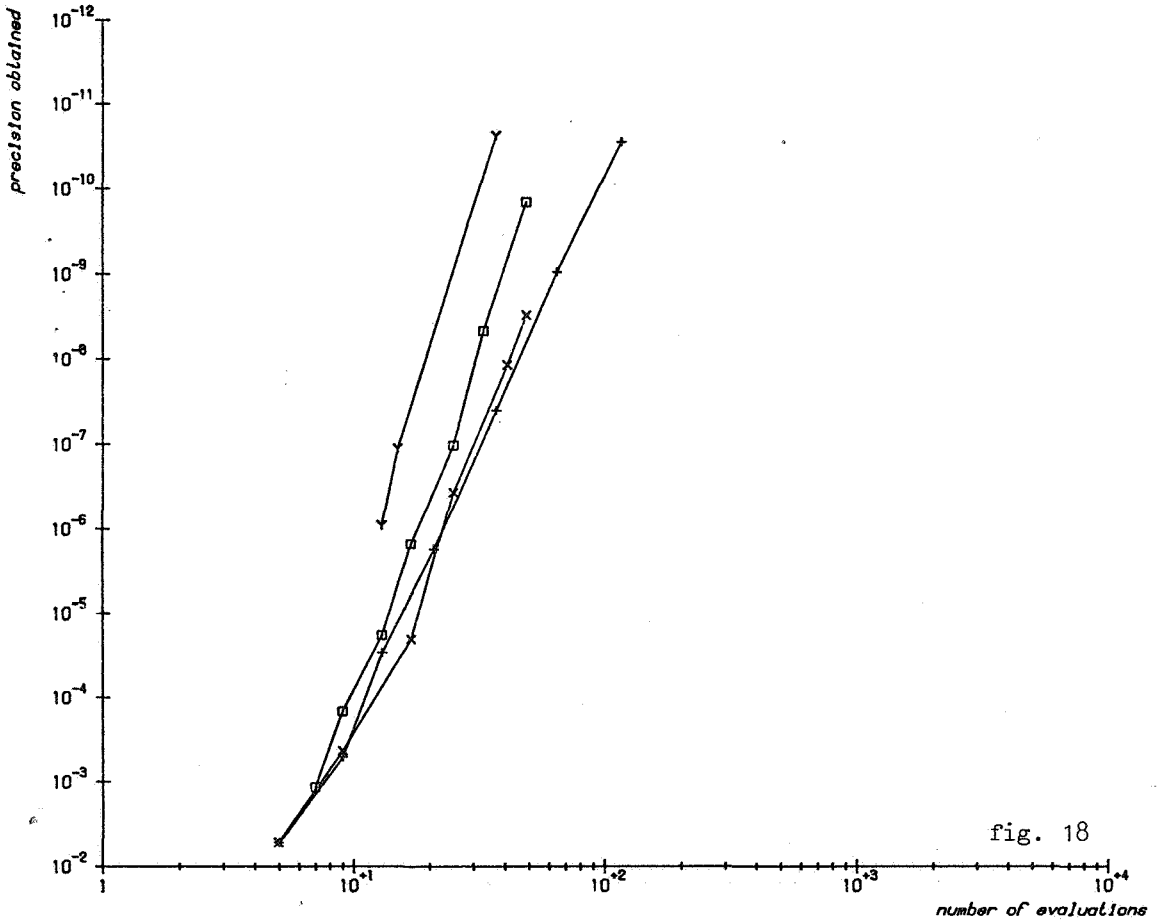
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 18

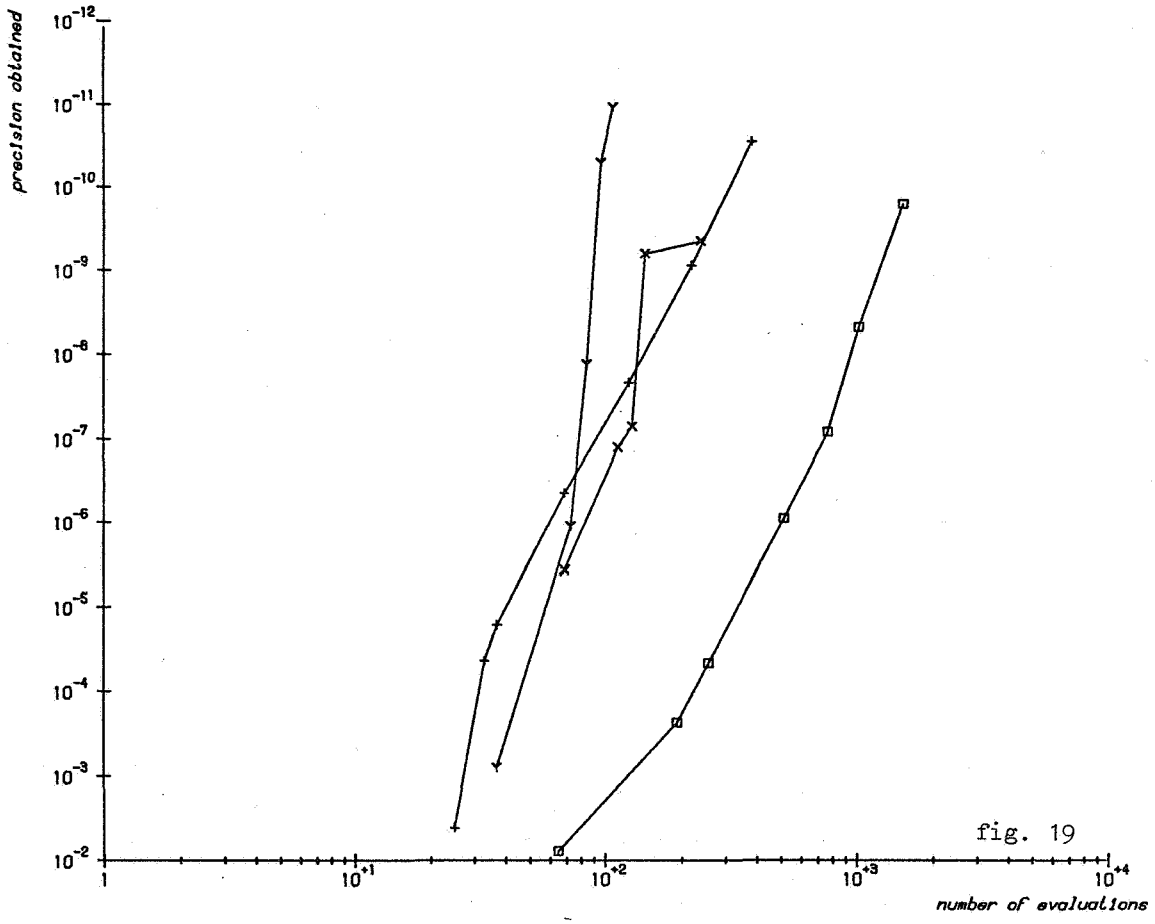
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 19

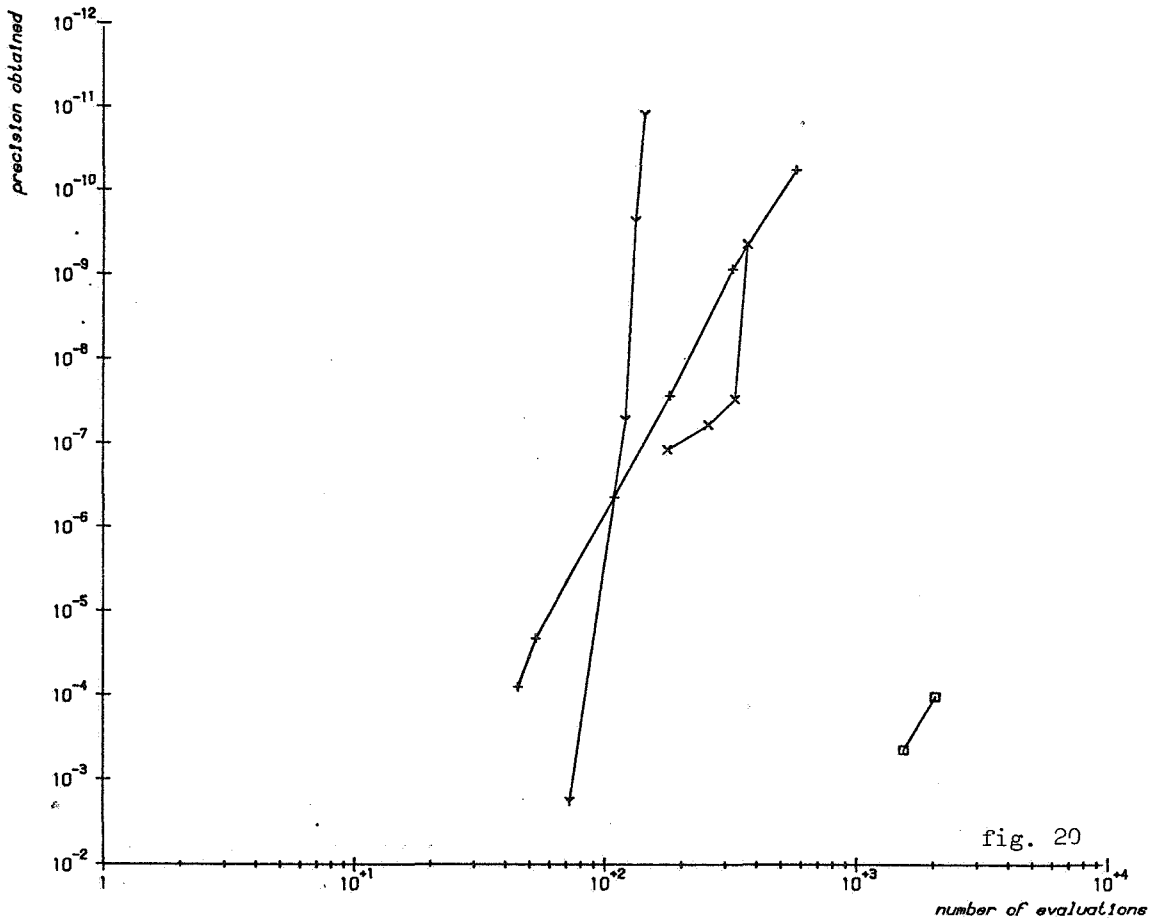
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 20

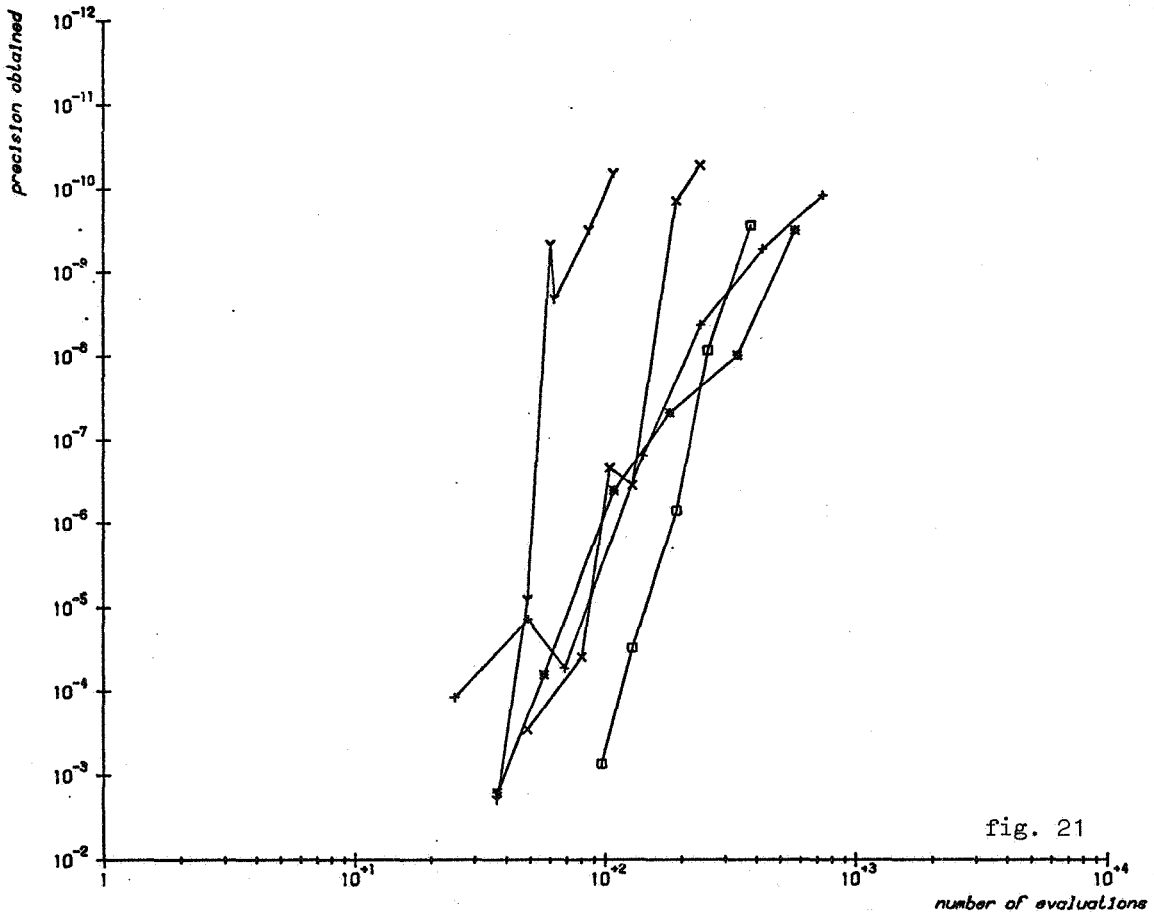
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 21

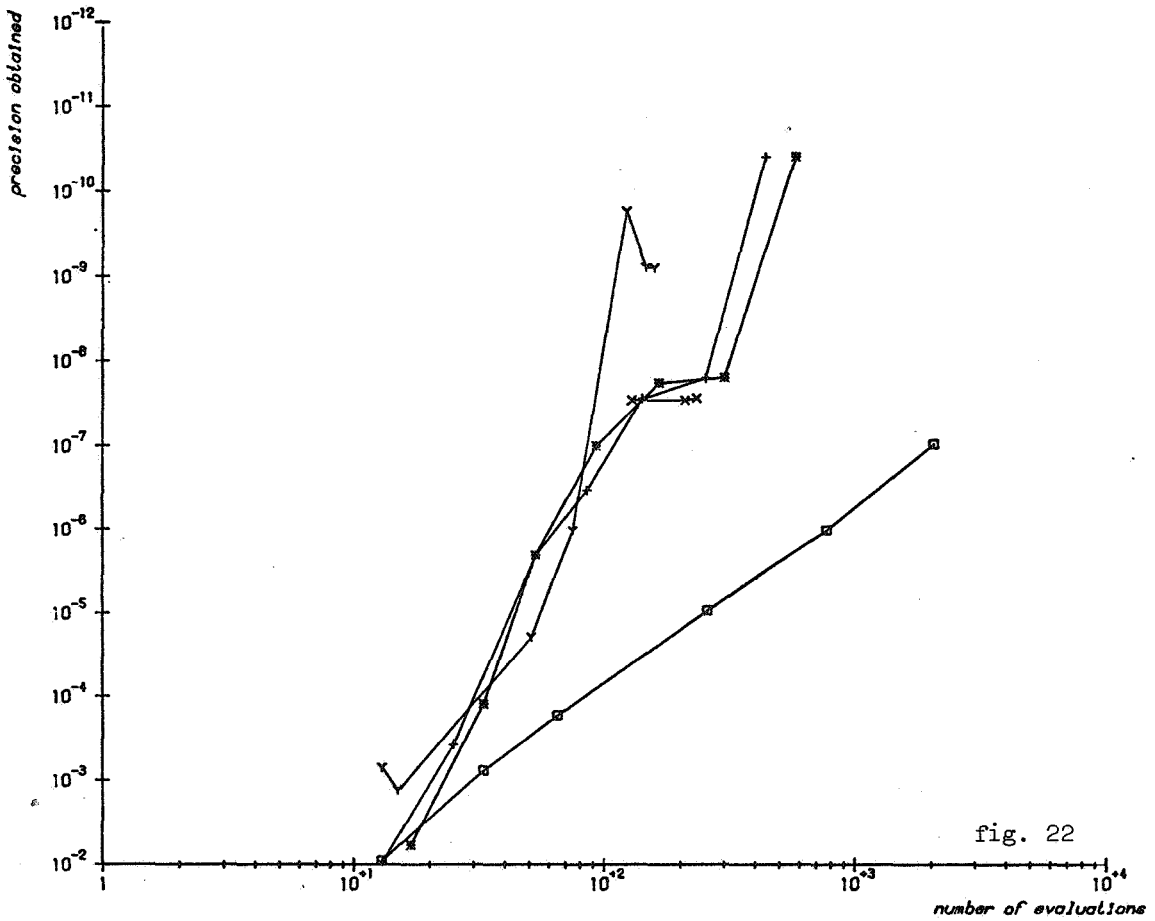
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 22

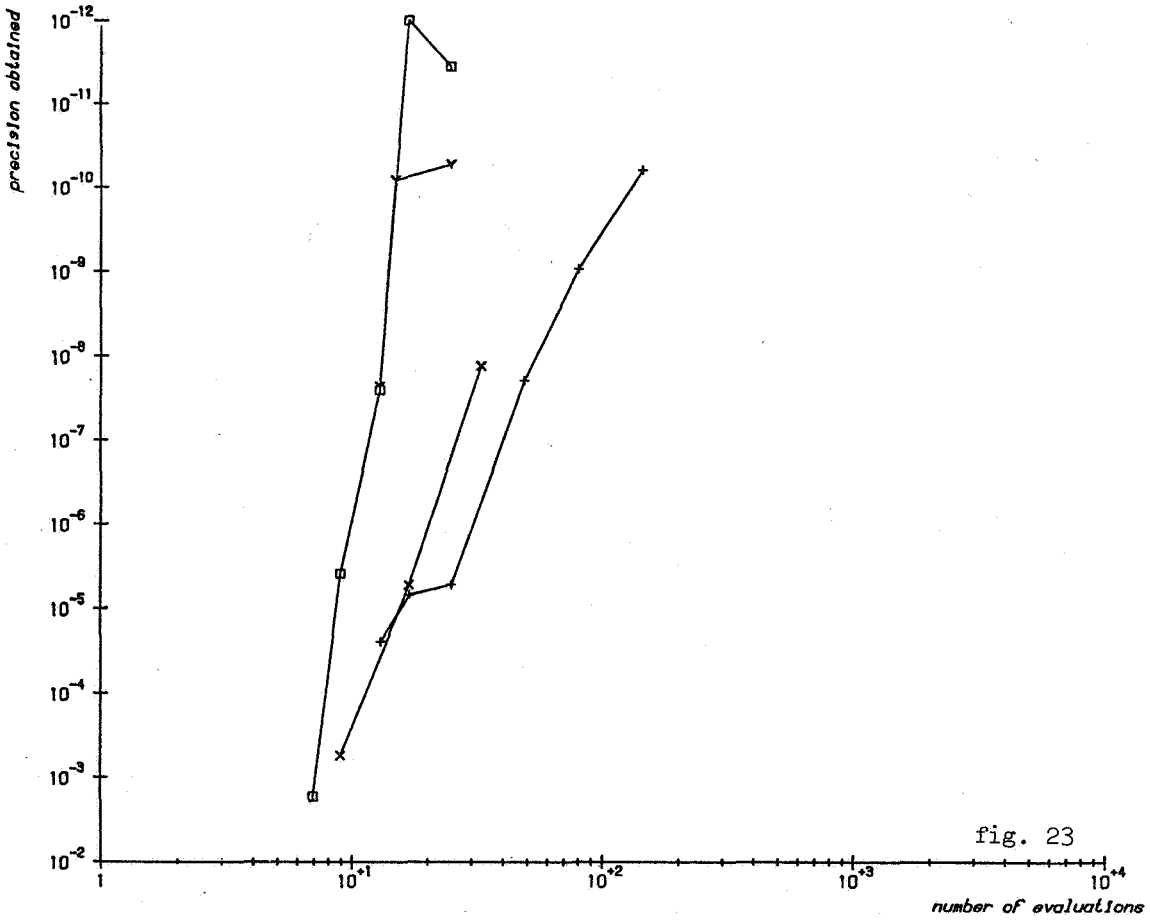
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 23

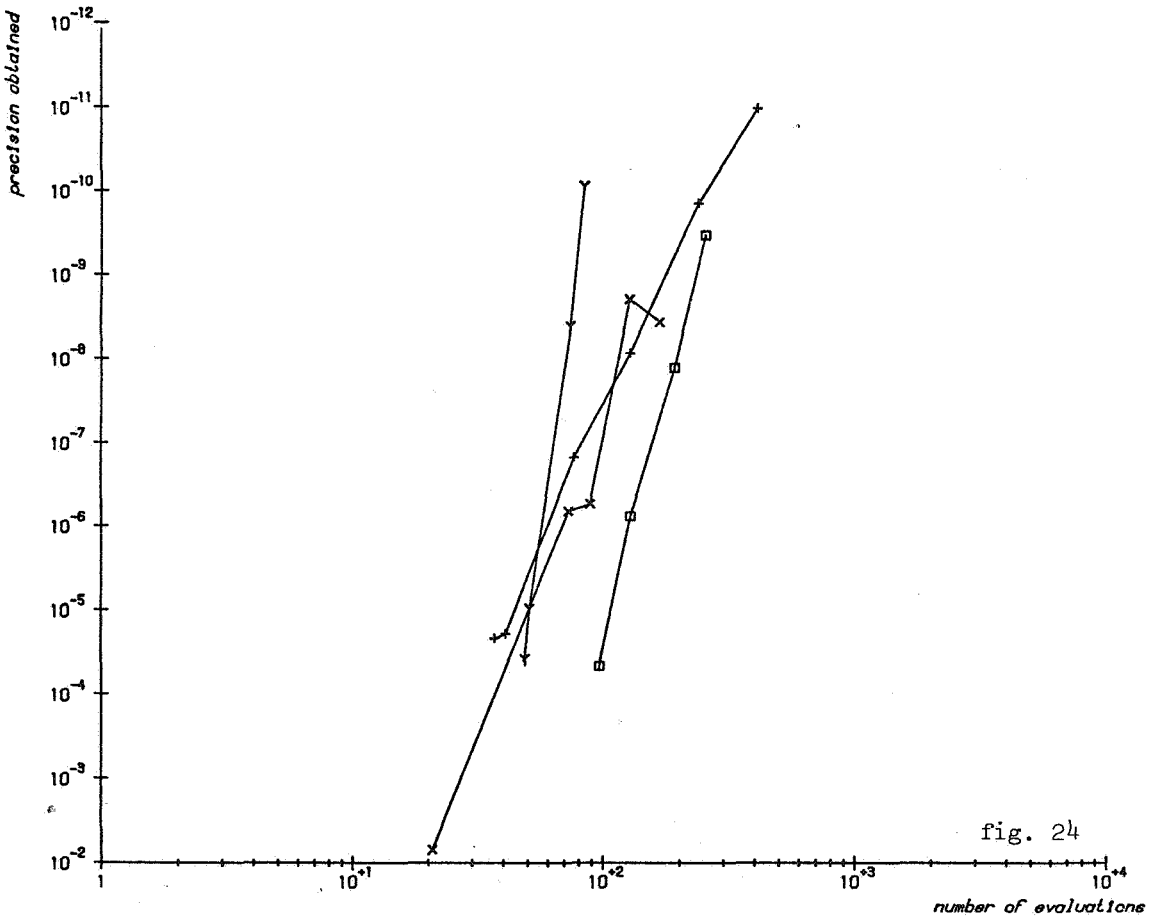
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 24

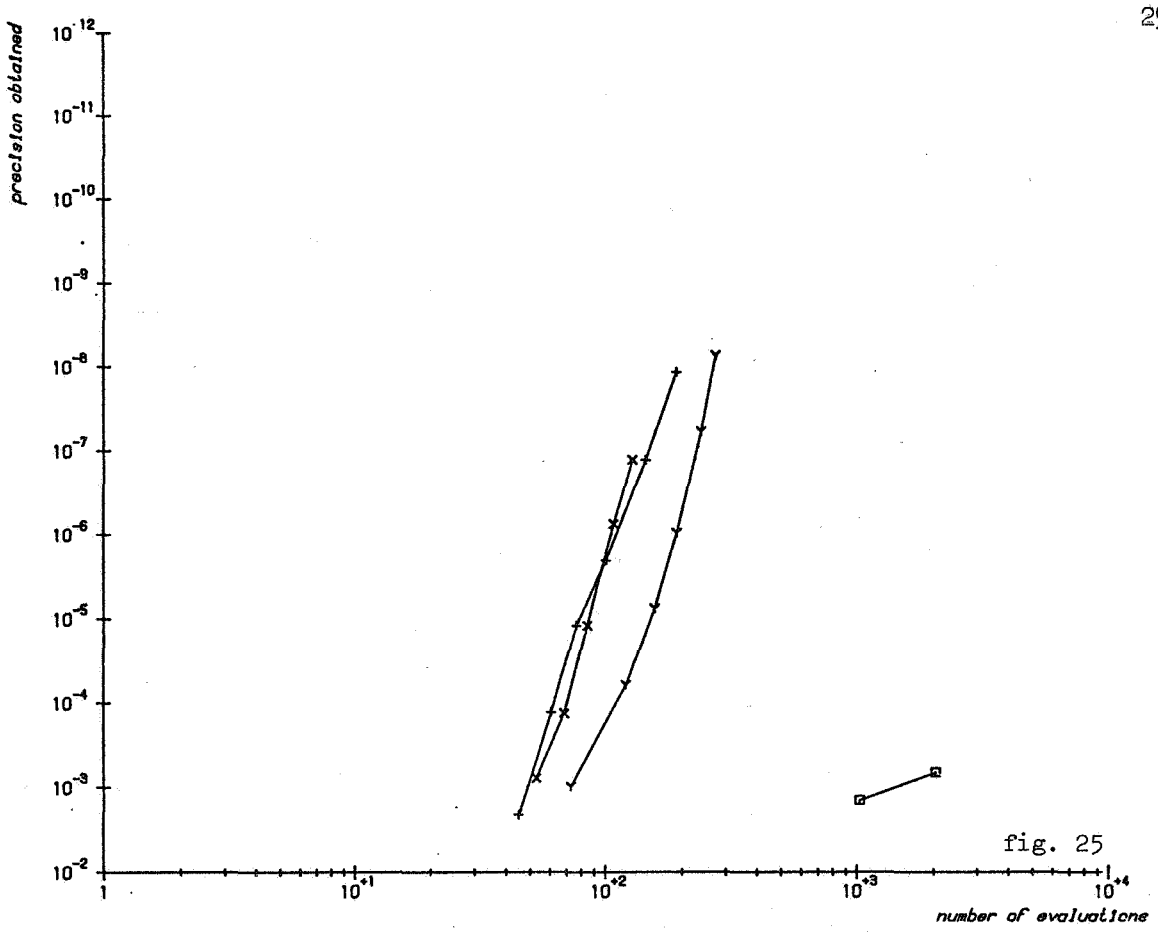
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 25

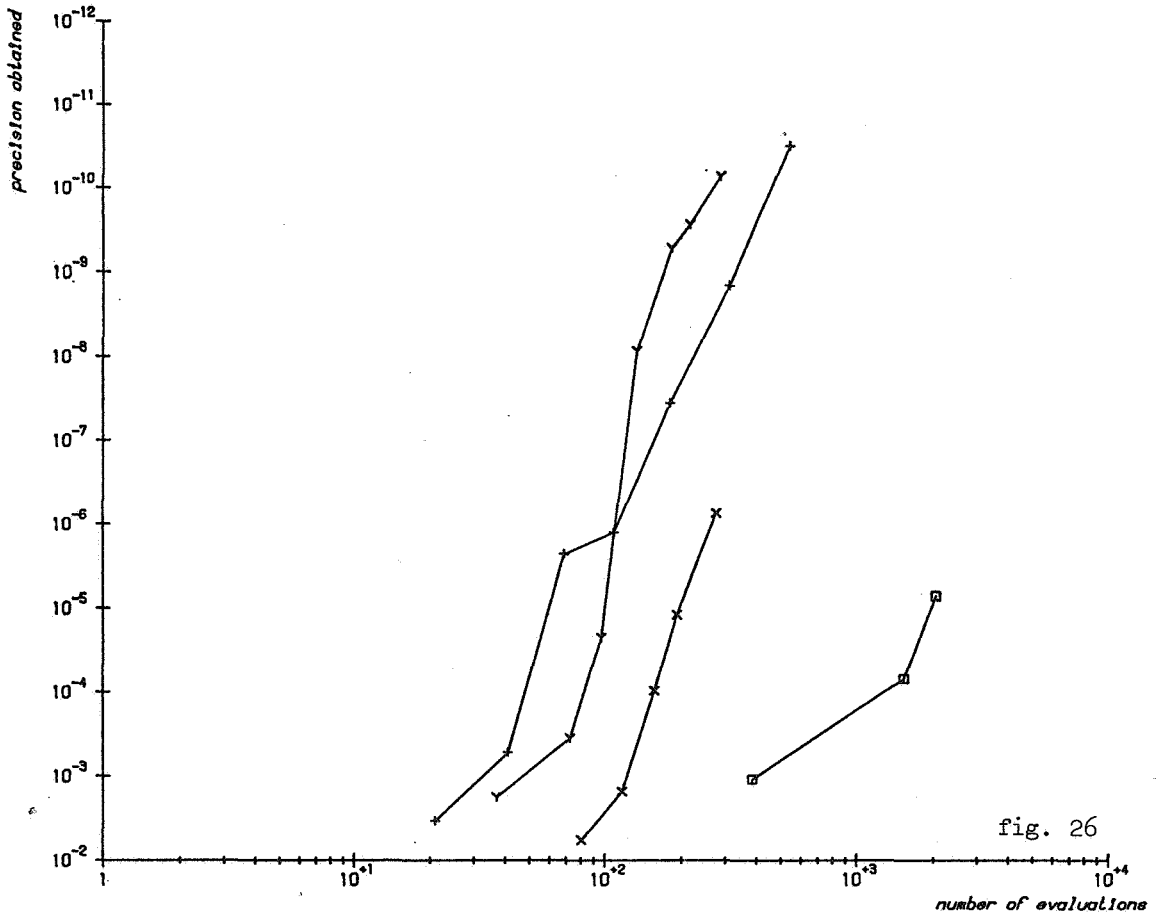
precision obtained
10^{-12}
10^{-11}
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
1
10^{+1}
10^{+2}
10^{+3}
10^{+4}
number of evaluations

fig. 26

5. Remarks

The marks of the procedure integral are plotted only for the integrals 6, 7, 8, 9, 10, 11, 17 and 21. For the other integrals the results of integral are (nearly) the same as the results of qad.
In the table below, the following abbreviations are used:

no marks: because the obtained precision is less then 10^{-2}, no marks are plotted for the tolerance(s)

same marks: because there is no difference between the obtained precision, only one mark is plotted for the tolerances

max: trapex needs the maximal number of subdivisions of the integration interval for the tolerances.

number of the integral	qad	int	qadrat	integral	trapex
1		no marks $10^{-2}, 10^{-3}$	no marks 10^{-2}		
2		no marks $10^{-2}, 10^{-3}$	no marks 10^{-2}		max 10^{-8}
3		no marks $10^{-2}, 10^{-3}$	no marks 10^{-2}		max $10^{-7}, 10^{-8}$
4	same marks $10^{-2}, 10^{-3}, 10^{-4}$	same marks $10^{-2}, 10^{-3}, 10^{-4}$	same marks $10^{-2}, \ldots, 10^{-8}$	same marks $10^{-2}, 10^{-3}, 10^{-4}$	same marks $10^{-3}, 10^{-4}$ same marks $10^{-5}, 10^{-6}$
5	same marks $10^{-2}, 10^{-3}$	same marks $10^{-2}, 10^{-3}$ same marks $10^{-4}, 10^{-5}$ same marks $10^{-6}, 10^{-7}$	same marks $10^{-2}, \ldots, 10^{-6}$ same marks $10^{-7}, 10^{-8}$	same marks $10^{-2}, 10^{-3}$	same marks $10^{-4}, 10^{-5}$ same marks $10^{-7}, 10^{-8}$
6	same marks $10^{-2}, 10^{-3}$	no marks $10^{-2}, 10^{-3}$ same marks $10^{-5}, 10^{-6}$	same marks $10^{-2}, \ldots, 10^{-5}$		same marks $10^{-2}, 10^{-3}, 10^{-4}$ same marks $10^{-5}, 10^{-6}$ same marks $10^{-7}, 10^{-8}$
7	no marks 10^{-2}	no marks $10^{-2}, 10^{-3}$	no marks 10^{-2} same marks $10^{-7}, 10^{-8}$		no marks $10^{-2}, 10^{-3}$ max $10^{-6}, 10^{-7}, 10^{-8}$
8	no marks $10^{-2}, 10^{-3}$	no marks $10^{-2}, 10^{-3}, 10^{-4}$	no marks $10^{-2}, 10^{-3}$ same marks $10^{-6}, 10^{-7}$	no marks 10^{-2}	no marks $10^{-2}, \ldots, 10^{-8}$ max $10^{-3}, \ldots, 10^{-8}$
9	no marks $10^{-2}, 10^{-3}, 10^{-4}$	no marks $10^{-2}, \ldots, 10^{-5}$	no marks $10^{-2}, 10^{-3}, 10^{-4}$ same marks $10^{-6}, 10^{-7}$	no marks $10^{-2}, 10^{-3}$	no marks $10^{-2}, \ldots, 10^{-8}$ max $10^{-4}, \ldots, 10^{-8}$
10	no marks $10^{-2}, \ldots, 10^{-6}$	no marks $10^{-2}, \ldots, 10^{-8}$	no marks $10^{-2}, \ldots, 10^{-6}$	no marks $10^{-2}, \ldots, 10^{-6}$	no marks $10^{-2}, \ldots, 10^{-8}$ max $10^{-6}, 10^{-7}$
11	no marks $10^{-2}, 10^{-3}, 10^{-4}$	no marks $10^{-2}, \ldots, 10^{-6}$	no marks 10^{-2}	no marks $10^{-2}, \ldots, 10^{-4}$	no marks $10^{-2}, \ldots, 10^{-8}$ max $10^{-4}, \ldots, 10^{-8}$
12	no marks $10^{-2}, 10^{-3}$	no marks $10^{-2}, 10^{-3}$	same marks $10^{-2}, 10^{-3}$ same marks $10^{-4}, 10^{-5}$ same marks $10^{-6}, 10^{-7}$	no marks $10^{-2}, 10^{-3}$	same marks $10^{-6}, 10^{-7}$

number of the integral	qad	int	qadrat	integral	trapex
13	no marks $10^{-2},10^{-3}$	no marks $10^{-2},10^{-3},10^{-4}$ same marks $10^{-6},10^{-7}$	same marks $10^{-2},10^{-3}$ same marks $10^{-4},10^{-5}$ same marks $10^{-6},10^{-7}$	no marks $10^{-2},10^{-3}$	same marks $10^{-6},10^{-7}$
14	no marks $10^{-2},10^{-3}$	no marks $10^{-2},10^{-3}$	same marks $10^{-2},10^{-3}$	no marks $10^{-2},10^{-3}$	no marks 10^{-2} max $10^{-7},10^{-8}$
15	no marks 10^{-2}	no marks $10^{-2},10^{-3}$		no marks 10^{-2}	no marks 10^{-2} max $10^{-6},10^{-7},10^{-8}$
16	no marks 10^{-2}	no marks 10^{-2}		no marks 10^{-2}	no marks 10^{-2} max $10^{-5},\ldots,10^{-8}$
17	no marks 10^{-2}	no marks 10^{-2} same marks $10^{-3},10^{-4}$	no marks 10^{-2}		max $10^{-7},10^{-8}$
18		same marks $10^{-3},10^{-4}$	same marks $10^{-2},\ldots,10^{-5}$ no marks $10^{-7},10^{-8}$		
19		no marks $10^{-2},10^{-3}$	same marks $10^{-3},10^{-4}$ same marks $10^{-6},10^{-7}$		
20	no marks 10^{-2}	no marks $10^{-2},10^{-3},10^{-4}$	no marks 10^{-2} same marks $10^{-4},10^{-5}$ same marks $10^{-7},10^{-8}$	no marks 10^{-2}	no marks 10^{-2} max $10^{-4},\ldots,10^{-8}$
21		no marks 10^{-2}	same marks $10^{-3},10^{-4}$	no marks 10^{-2}	same marks $10^{-5},10^{-6}$ same marks $10^{-7},10^{-8}$
22		no marks $10^{-2},\ldots,10^{-5}$			max $10^{-7},10^{-8}$

number of the integral	qad	int	qadrat	integral	trapex
23	no marks 10^{-2}	no marks 10^{-2} same marks $10^{-4}, 10^{-5}$ same marks $10^{-6}, 10^{-7}, 10^{-8}$	same marks $10^{-2}, \ldots, 10^{-5}$ same marks $10^{-6}, 10^{-7}$	no marks 10^{-2}	same marks $10^{-4}, 10^{-5}$ same marks $10^{-6}, 10^{-7}$
24	no marks 10^{-2}	no marks 10^{-2} same marks $10^{-6}, 10^{-7}$	no marks 10^{-2} same marks $10^{-6}, 10^{-7}, 10^{-8}$	no marks 10^{-2}	no marks 10^{-2} same marks $10^{-5}, 10^{-6}$ same marks $10^{-7}, 10^{-8}$
25	no marks 10^{-2}	no marks $10^{-2}, 10^{-3}$	no marks 10^{-2}	no marks 10^{-2}	no marks 10^{-2} max $10^{-4}, \ldots, 10^{-8}$
26		no marks $10^{-2}, 10^{-3}$			no marks 10^{-2} max $10^{-5}, \ldots, 10^{-8}$

Considering the figures and the table, it is obvious that there is only a small difference between the results of the procedures qad and integral; for the sake of clearness the marks of integral had to be omitted in most cases.
It is not surprising that with a few exceptions trapex delivers bad results. The procedure is non - adaptive, so that it is useful only for smooth integrands.
The procedure qadrat, which makes use of 11, 13 and 15 - point formulas (order 12, 14 and 16), is very useful for high precision.