

```

> restart;
>
> steffensen :=
proc(g::algebraic,p::numeric,tol::positive,no::posint,root::name)
local G, OK, P0, TOL, NO, FLAG, I, P1, P2, D, P, GP;
# make sure initial approximation and tolerance are of type floating
point
P0:=evalf(p);
TOL:=evalf(tol);
# Make G into a function
G:=unapply(g,x);
NO:=no;
printf(` k          p[0](k)          p[1](k)
p[2](k)\n`);
printf(` -          -          -
-----\n`);
# execute the algorithm
> I := 1;
> OK := TRUE;
FLAG:=2;
> while I <= NO and OK = TRUE do
> P1 := evalf(G(P0));
> P2 := evalf(G(P1));
> if abs(P2-2*P1+P0) < 1.0e-20 then
FLAG := 1;
> D := G(P0);
> printf(`\n\nDenominator = 0, method fails\n`);
> printf(`Best possible solution is %a`,args[5]);
printf(` = %20.15f \n`,P2);
printf(`with g(%a`,args[5]);
printf(`) = %20.15f\n`,D);
RETURN();
> OK := FALSE;
> else
> D := (P1-P0)*(P1-P0)/(P2-2*P1+P0);
> fi;
> P := P0-D;
> if FLAG = 2 then
> printf(`%3d%23.15e%23.15e%23.15e\n`, I, P0, P1, P2);
> fi;
> if abs(D) < TOL then
GP:=G(P);
> printf(`%3d%23.15e\n`, I+1, P);
printf(`\nThe approximate solution is %a`,args[5]);
printf(` = %23.15f \n`,P);
printf(`with  $\alpha$ (%a`,args[5]);

```

```

printf(`) = %23.15f\n`,GP);
root:=P;
> OK := FALSE;
> else
> I := I+1;
> P0 := P;
> fi;
od;
> if OK = TRUE then
GP:=G(P);
> printf(`\nIteration number %3d`,NO);
printf(` gave approximation %23.15f\n`,P);
printf(`G(P) = %23.15f not within tolerance %23.15f\n`,GP,TOL);
RETURN();
else
P;
fi;
end;

```

Warning, imaginary unit `I` used as a local variable

steffensen := **proc**(g::algebraic, p::numeric, tol::positive, no::posint, root::name)

local G, OK, P0, TOL, NO, FLAG, I, P1, P2, D, P, GP;

P0 := evalf(p);

TOL := evalf(tol);

G := unapply(g, x);

NO := no;

printf(` k p[0](k) p[1](k) p[2](k));

printf(` - ----- ----- -----);

I := 1;

OK := TRUE;

FLAG := 2;

while I <= NO and OK = TRUE **do**

P1 := evalf(G(P0));

P2 := evalf(G(P1));

if `abs`(P2 - 2*P1 + P0) < 1.0*10⁻²⁰ **then**

FLAG := 1;

D := G(P0);

printf(` Denominator = 0, method fails);

printf(` Best possible solution is %a`, args[5]);

printf(` = %20.15f ; P2);

printf(` with g(%a`, args[5]);

printf(` = %20.15f ; D);

RETURN();

OK := FALSE

else

D := (P1 - P0^2)/(P2 - 2*P1 + P0)

end if

P := P0 - D;

if FLAG = 2 **then** printf(`%3d%23.15e%23.15e%23.15e` ; I, P0, P1, P2) **end if**

if `abs`(D) < TOL **then**

GP := G(P);

printf(`%3d%23.15e` ; I + 1, P);

printf(` The approximate solution is %a`, args[5]);

printf(` = %23.15f ; P);

```

        printf(`with g(%a`, args[5]);
        printf(` = %23.15f`, GP);
        root := P;
        OK := FALSE
    else
        I := I + 1;
        P0 := P
    end if
end do
if OK = TRUE then
    GP := G(P);
    printf(` Iteration number %3d`, NO);
    printf(` gave approximation %23.15f`, P);
    printf(` G(P) = %23.15f not within tolerance %23.15f`, GP, TOL);
    RETURN()
else
    P
end if
end proc

```

```

> steffensen_dir:=proc()
printf(`steffensen returns a root a function of the form x=g(x).\n`);
printf(`It allows up to 15 decimal places.\n\n`);
printf(`The arguments for steffensen are:\n`);
printf(`(1)function expression in x\n`);
printf(`(2)initial approximation\n`);
printf(`(3)tolerance\n`);
printf(`(4)maximum number of iterations\n`);
printf(`(5)variable for returning root\n\n`);
printf(`If assigning the result to a variable, have the\n`);
printf(`variable and the 5th argument the same.\n\n`);
printf(`If r is the variable for returning the root\n`);
printf(`and has already been given a value,\n`);
printf(`the procedure should be preceded by the statement:\n`);
printf(`r:='r'`);
end;

```

```

steffensen_dir := proc()
    printf(`steffensen returns a root a function of the form x=g(x).`);
    printf(`It allows up to 15 decimal places.`);
    printf(`The arguments for steffensen are:`);
    printf(`(1)function expression in x`);
    printf(`(2)initial approximation`);
    printf(`(3)tolerance`);
    printf(`(4)maximum number of iterations`);
    printf(`(5)variable for returning root`);
    printf(`If assigning the result to a variable, have the`);
    printf(`variable and the 5th argument the same.`);
    printf(`If r is the variable for returning the root`);
    printf(`and has already been given a value,`);
    printf(`the procedure should be preceded by the statement:`);
    printf(`r:='r'`)
end proc

```

