

GENAPAC

A genetic algorithm for parameter calibration

User's manual
Version 1.0

Sébastien Schneider

September, 2010

SCK•CEN
Boeretang 200
BE-2400 Mol
Belgium

PAS

GENAPAC

A genetic algorithm for parameter calibration

User's manual
Version 1.0

Sébastien Schneider

September, 2010
Status: Unclassified
ISSN 1782-2335

SCK•CEN
Boeretang 200
BE-2400 Mol
Belgium

© SCK•CEN
Studiecentrum voor Kernenergie
Centre d'étude de l'énergie Nucléaire
Boeretang 200
BE-2400 Mol
Belgium

Phone +32 14 33 21 11
Fax +32 14 31 50 21

<http://www.sckcen.be>

Contact:
Knowledge Centre
library@sckcen.be

RESTRICTED

All property rights and copyright are reserved. Any communication or reproduction of this document, and any communication or use of its content without explicit authorization is prohibited. Any infringement to this rule is illegal and entitles to claim damages from the infringer, without prejudice to any other right in case of granting a patent or registration in the field of intellectual property.

SCK•CEN, Studiecentrum voor Kernenergie/Centre d'Etude de l'Energie Nucléaire
Stichting van Openbaar Nut – Fondation d'Utilité Publique - Foundation of Public Utility

Registered Office: Avenue Herrmann Debroux 40 – BE-1160 BRUSSEL

Operational Office: Boeretang 200 – BE-2400 MOL

Contents

1	Introduction	7
2	Description of the computer code	8
2.1	<i>Features of GENAPAC</i>	8
2.2	<i>Source code</i>	8
2.3	<i>Resource files</i>	9
2.4	<i>Output files</i>	10
3	Providing source code and resource files	11
4	Examples	26
4.1	<i>Calibration of unsaturated hydraulic parameters of concrete</i>	26
4.2	<i>Calibration of unsaturated hydraulic parameters for a heterogeneous soil profile</i>	28
5	Conclusion	32
6	Acknowledgement.....	32
7	References	32

The GENAPAC computer code is a genetic algorithm for parameter calibration. The code is written in FORTRAN 95. This user's manual describes the source code and provides information to facilitate its use. This document contains an introduction (Chapter 1) which gives an overview of the general functioning of genetic algorithms. Chapter 2 provides explanations about the main program and the subroutines, whereas the Chapter 3 provides the source code. Two examples are given to demonstrate the versatility of the code in determining parameter estimates for complex problems (Chapter 4). The GENAPAC code can be coupled with any computer code such as those that solve the water flow and contaminant transport equations in soils and aquifers. For both of the two examples the HYDRUS-1D code (Šimůnek et al., 2009) for variably saturated water flow was used.

1 Introduction

Heuristic search algorithms are efficient techniques for solving complex optimization problems. One of such algorithms is the genetic algorithm (GA). General literature on genetic algorithms is available from Goldberg (1989), or Goldberg and Deb (1991), amongst others. Genetic Algorithms (GAs) are stochastic search procedures widely used in many sciences for complex non-linear optimization problems (Zhang et al., 2009; Ines and Mohanty, 2008). GAs are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, selection, crossover and mutation. To identify the optimal in a specified search space, GAs are implemented in a computer algorithm by a population of abstract representations (called chromosomes) of candidate solutions (called individuals) which evolve gradually towards an optimal solution of the optimization problem. Solutions could be represented in binary form as strings of 0s and 1s, but for many applications in soil hydrology, a representation with real-parameter values is much more convenient. The evolutionary process starts from a population of randomly generated individuals and proceeds through successive generations (see the general flowchart in Figure 1). In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness, i.e. objective criterion), and modified (recombined and possibly randomly mutated) to form a new population. This loop is repeated in an iterative process, until the algorithm terminates when either a maximum number of generations has been produced or when some predefined performance criterion has been met.

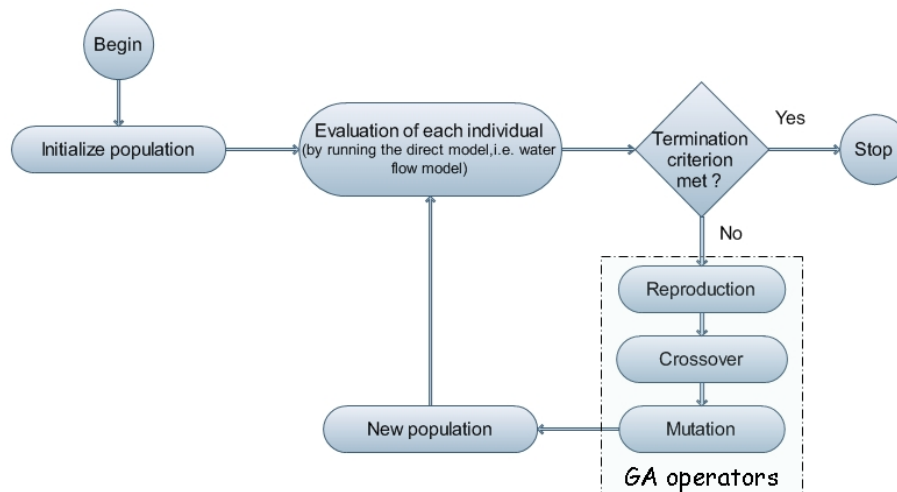


Figure 1. Flowchart of the principle of the genetic algorithm.

2 Description of the computer code

2.1 Features of GENAPAC

GENAPAC is a genetic algorithm which has the following features:

- a) Real-parameters-coded.
- b) Selection operator: Roulette wheel selection.

The selection process is a key step in the GA that embodies the idea of “fittest survival” in nature. A fitness level is used to associate a probability of selection for each individual (=solution). If FF_i is the fitness of individual i in the population, its probability of being selected, p_i , is:

$$p_i = \frac{1}{FF_i \sum_{j=1}^M FF_j} \quad (1)$$

where M is the number of individuals in the population. The selection process is similar to a roulette-wheel: a proportion of the wheel is assigned to each individual based on its fitness value. This is achieved by dividing the fitness of an individual by the total fitness of all individuals, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated. Thus each solution has a non-zero probability of being selected, the solutions with highest fitness being more likely selected.

- c) Cross-over operator: multiple points cross-over
- d) Mutation operator: non-fixed mutation strength
- e) Elitism operator: 1 elite saved per generation and per island
- f) Migration operator: 1 elite per island duplicates in any other island

2.2 Source code

Source code files are the following:

- GENAPAC.f (main program)
- newgen.f (subroutine)
- random.f (subroutine)
- localsearch.f (subroutine)
- output.f (subroutine)
- func.f (subroutine)

GENAPAC.f is the main program which contains the calls to the subroutines.

newgen.f is responsible for the creation of the new generations. It contains subroutines that implement the following operators: selection, cross-over, mutation, elitism and migration.

random.f is a subroutine which creates random numbers.

localsearch.f is a subroutine which allows the user to perform a local search after the genetic algorithm has finished. The algorithm starts with the best solution found by the genetic algorithm and tries to obtain better solutions by diving each parameter range in a dichotomous process. Parameters are processed sequentially. The algorithm terminates when no further improvement in the solution is found.

output.f is a subroutine responsible for the creation of output files. Some details about those files will be given in the next section.

func.f is a subroutine responsible for the calculation of the objective function. Any kind of problem can be treated with this algorithm provided that the objective function is defined in this subroutine. The user should define is own objective function in this subroutine, paying attention that:

- the parameters are defined in the vector *paramer(npam,n)* where *npam* is the number of parameters to optimize and *m* the numbers of islands.
- after being calculated, the objective function has to be recorded in the variable *fx*.

2.3 *Resource files*

Two resource files are necessary:

- config.dat
- parameter.inc

config.dat is the file in which the lower and upper bounds are provided for each parameter.

parameter.inc is a file which allows the user to parameterize GENAPAC. The following variables have to be defined:

- *npam*: number of parameters to be optimized.
- *pop*: number of individual in the population, or in each subpopulation if the migration operator is used.
- *maxgen*: number of generations after which the algorithm terminates.
- *cont*: number of islands. Set *cont*=1 if no migration operator is wanted.
- *elitism*: setting 1 enables the elitism operator. Any other value disables the elitism operator.
- *mfreq*: set the frequency of the mutation operator. For instance *mfreq*=0.2 leads to a mutation operator acting on 20% of the parameters. Therefore *mfreq* should always be between 0.0 and 1.0.

- *mint*: defines how the intensity of the mutation evolves during the algorithm. Intensity represents the mean distance from the original parameter value to its value after mutation. For instance, it might be useful to have a strong mutation factor at the beginning of the algorithm (in order to explore the parameter search space) and a weak one at the end of the algorithm (since it is not needed anymore to explore the whole parameter search space). *mint* allows the user to apply a progressive increase or decrease of the mutation intensity during the optimization process. A negative value of *mint* increases the intensity of mutation with time whereas a positive value of *mint* decreases the intensity of mutation. For a fixed intensity of the mutation set *mint*=0. Recommended value belongs to the interval [0.0;1.0].
- *locals*: set 1 for enabling the local search; any other value disables this additional search.

The variables used are referenced in Table 1. A short description of each variable is given as well.

Table 1. Name and description of the variables.

File	Variable	Type	Description
parameter.inc	nparam	integer	number of parameters
	pop	integer	size of the population
	maxgen	integer	maximum number of generations
	cont	integer	number of islands
	mix	integer	period of isolation (between two phases of immigration)
	elitism	integer	enable elitism (if equal to 1)
	mfreq	real	frequency of the mutation operator
	mint	real	intensity of mutation
GENAPAC.f	gen	integer	generation
	fx	real	store the value of the objective function
	parlim	real	contains the bounds for each parameter
	param	real	parameter values of the candidate solution
	generation	real	store all tested solutions (parameter and objective function values)
	newgener	real	contains the solutions which will be tested in the next generation
newgen.f	individu	integer	refers to the first parent
	individu2	integer	refers to the second parent
	gene	integer	considered gene for cross-over
output.f	fitness	real	average fitness of a specific generation

2.4 Output files

Two files are provided in order to summarize the optimization results:

- **Generations.out**: contains for each generation the different parameter sets which have been evaluated and ranked from the better to the worst solution (the first column corresponds to the objective function values whereas the other columns indicate the parameter values).
- **Results.out**: contains for each generation (and for each island if the migration operator is enabled) the best solution found. In addition, if a local search is

performed after the GA is terminated, it contains also the best value of the objective function found for each iteration of the local search.

3 Providing source code and resource files

In this section the source code and the resource files are provided. The **func.f** file gives an example of a hydraulic parameter optimization problem for a concrete sample. This example aims at estimating unsaturated soil hydraulic parameters using two types of experimental data:

- water retention data,
- capillary absorption data.

Details on the experimental device are provided in Schneider et al. (2010). The HYDRUS-1D package (Šimůnek et al., 2009) is used for simulating unsaturated water flow. Therefore the **func.f** file contains instructions for modifying files in the HYDRUS-1D input files, for reading HYDRUS-1D output files, and for executing HYDRUS-1D from the GA.

GENAPAC.f

```

program GENAPAC

implicit none
include 'parameter.inc'
character *500 lignes(3000)
integer i,j,k,l,m,gen,idum
real ran
double precision fx,parlim(nparam,2),generation(pop,nparam,cont)
&,param(nparam,cont),fitness(pop,nparam+1,cont),Verybest(nparam+1)
&,fit(pop,nparam+1,cont),newgener(pop,nparam,cont),
&stockbestOF(nparam+1,cont)

open(11,file='C:\ABSORPTION\fortran_sources\
&config.dat',status='old')
open(13,file='C:\ABSORPTION\Fs\observation.dat',status='old')
open(92,file='C:\ABSORPTION\Fs\BESTresinv.out',status='unknown')
open(93,file='C:\ABSORPTION\Fs\BESTparam.out',status='unknown')
open(96,file='C:\ABSORPTION\Fs\Generations.out',status='unknown')
open(98,file='C:\ABSORPTION\Fs\Results.out',status='unknown')

!Reading parameter bounds =====

read(11,'(A)') lignes(1)
do i=1,nparam
read(11,*) parlim(i,1),parlim(i,2)
enddo

!File Initialization =====

write(93,*) '99999999.99';write(96,*) ' '
write(98,'(a,$)') ' Gen '
do i=1,cont
write(98,'(a,$)') ' Cont ';write(98,'(i1,$)') i;
write(98,'(a,$)') ' '
enddo
write(98,*) ' '

!Starting with first generation =====

gen=1

```

```

do m=1,cont
do i=1,pop
do j=1,nparam
call rand(idum,ran)
generation(i,j,m)=ran*(parlim(j,2)-parlim(j,1))+parlim(j,1)
enddo
enddo
enddo

do m=1,cont
do i=1,pop
call selector(generation,fitness,param,nparam,pop,i,m,cont)
call func(param,nparam,fx,m,cont)
fitness(i,1,m)=fx
do k=1,nparam
fitness(i,k+1,m)=generation(i,k,m)
enddo
enddo
call ranking(fitness,fit,pop,nparam,m,cont)
call stock(fit,pop,nparam,stockbestOF,Verybest,m,cont)
call output(gen,maxgen,generation,nparam,pop,parlim,fit,m,cont)
enddo

!Next generations =====

do gen=2,maxgen
rewind(15)

do m=1,cont
call newgen(fit,pop,nparam,newgener,maxgen,gen
&,generation,parlim,m,cont,mint,mfreq)
enddo

if (mod(gen,mix).EQ.0) then
call continent(generation,pop,nparam,cont,stockbestOF)
endif

do m=1,cont
do i=1,pop
call selector(generation,fitness,param,nparam,pop,i,m,cont)
call func(param,nparam,fx,m,cont)
fitness(i,1,m)=fx
do k=1,nparam
fitness(i,k+1,m)=generation(i,k,m)
enddo
enddo

call ranking(fitness,fit,pop,nparam,m,cont)
if (elitism.EQ.1) then
call elitism(fit,pop,nparam,stockbestOF,elitism,m,cont)
endif
call stock(fit,pop,nparam,stockbestOF,Verybest,m,cont)
call output(gen,maxgen,generation,nparam,pop,
&parlim,fit,m,cont)
enddo
enddo

!Local search =====

IF (locals.EQ.1) THEN
call localsearch(fit,pop,param,parlim,nparam,fx,m,cont,
&Verybest)
ENDIF

!END =====

write (*,*) 'END'

close(11)

```

```

close(13)
close(92)
close(93)
close(96)
close(98)

end program

```

newgen.f

```

subroutine newgen(fit,pop,nparam,newgener,maxgen,gen
&,generation,parlim,m,cont,mint,mfreq)

implicit none
integer i,j,k,m,n,pop,nparam,idum,r1,r2,maxgen,gen,cont,
&individu,individu2,gene
real ran,harvest,maxi(pop),FxFOT,mint,mfreq
double precision fit(pop,nparam+1,cont),newgener(pop,nparam,cont)
&,generation(pop,nparam,cont),parlim(nparam,2)

!Giving weight to solutions =====

FxFOT=0.0
do i=1,pop
FxFOT=FxFOT+1/fit(i,1,m)
enddo
maxi=0.0
do i=1,pop
if (i.eq.1) then
maxi(i)=(1/fit(i,1,m))/FxFOT
else
maxi(i)=maxi(i-1)+(1/fit(i,1,m))/FxFOT
endif
enddo

!Cross-over =====

do i=1,pop
call rand(idum,ran)
do k=1,pop
individu=k
if (ran.LT.maxi(k)) exit
enddo
call rand(idum,ran)
do k=1,pop
individu2=k
if (ran.LT.maxi(k)) exit
enddo
do j=1,nparam
call rand(idum,ran)
if (ran.LT.0.5) then
newgener(i,j,m)=fit(individu,j+1,m)
else
newgener(i,j,m)=fit(individu2,j+1,m)
endif
enddo
enddo

!Mutation =====

do i=1,pop
do j=1,nparam
maxi(j)=real(j)/real(nparam)
enddo
do k=1,int(nparam*mfreq)
call rand(idum,ran)
do j=1,nparam
gene=j
if (ran.LT.maxi(j)) exit

```

```

        enddo

        call rand2(harvest)
        newgener(i, gene, m) = newgener(i, gene, m) *
& ((harvest+1) ** ((real(maxgen-gen)/real(maxgen)) ** mint))
        if (newgener(i, gene, m).LT.parlim(gene, 1)) then
            newgener(i, gene, m) = parlim(gene, 1)
        endif
        if (newgener(i, gene, m).GT.parlim(gene, 2)) then
            newgener(i, gene, m) = parlim(gene, 2)
        endif
    enddo
enddo

!New population =====

do i=1, pop
do j=1, nparam
    generation(i, j, m) = newgener(i, j, m)
enddo
enddo

end subroutine

!=====

subroutine continent(generation, pop, nparam, cont, stockbestOF)
implicit none
integer i, j, k, m, pop, nparam, cont
double precision stockbestOF(nparam+1, cont),
& generation(pop, nparam, cont)

do m=1, cont
do k=1, cont
do j=1, nparam
    generation(pop-k, j, m) = stockbestOF(j+1, k)
enddo
enddo
enddo
end subroutine

!=====

subroutine selector(generation, fitness, param, nparam, pop, i, m, cont)
implicit none
integer i, j, l, m, nparam, pop, cont
double precision param(nparam, cont), generation(pop, nparam, cont)
double precision fitness(pop, nparam+1, cont)

do l=1, pop
do j=1, nparam
    fitness(l, j+1, m) = generation(l, j, m)
enddo
enddo
do j=1, nparam
    param(j, m) = generation(i, j, m)
enddo

end subroutine

!=====

subroutine ranking(fitness, fit, pop, nparam, m, cont)

implicit none
integer i, j, k, m, pop, nparam, cont
double precision fit(pop, nparam+1, cont), fitness(pop, nparam+1, cont)
double precision Bestfit

do i=1, pop
    fit(pop, 1, m) = 1000000000.0

```

```

        enddo

do j=1,pop
    Bestfit=15000.0

    do i=1,pop
        if (fitness(i,1,m).LT.Bestfit) then
            Bestfit=fitness(i,1,m)
        endif
    enddo
    fit(j,1,m)=Bestfit
    do i=1,pop
        if (fitness(i,1,m).EQ.Bestfit) then
            do k=1,nparam
                fit(j,k+1,m)=fitness(i,k+1,m)
            enddo
            fitness(i,1,m)=2000000000.0
        exit
    endif
    enddo
enddo
end subroutine

!=====

subroutine stock(fit,pop,nparam,stockbestOF,Verybest,m,cont)

implicit none
integer i,m,pop,nparam,cont
double precision fit(pop,nparam+1,cont),stockbestOF(nparam+1,cont)
&,Verybest(nparam+1)

do i=1,nparam+1
    stockbestOF(i,m)=fit(1,i,m)
    Verybest(i)=fit(1,i,m)
enddo

end subroutine

!=====

subroutine elitisme(fit,pop,nparam,stockbestOF,elitism,m,cont)

implicit none
integer i,j,k,m,cont,pop,nparam,elitism
double precision fit(pop,nparam+1,cont),stockbestOF(nparam+1,cont)

if (stockbestOF(1,m).LT.fit(1,1,m)) then
do i=2,pop
    do j=1,nparam+1
        fit(pop+2-i,j,m)=fit(pop+1-i,j,m)
    enddo
enddo
do j=1,nparam+1
    fit(1,j,m)=stockbestOF(j,m)
enddo
endif

end subroutine

```

random.f

```

subroutine rand(idum,ran)

implicit none
integer, parameter :: K4B=selected_int_kind(9)
integer(K4B), intent(INOUT) :: idum
real :: ran

integer(K4B), parameter :: IA=16807,IM=2147483647,IQ=127773

```

```

&,IR=2836
  real, save :: am
  integer(K4B), save :: ix=-1,iy=-1,k
  if (idum <= 0 .or. iy < 0) then
    am=nearest(1.0,-1.0)/IM
    iy=ior(ieor(888889999,abs(idum)),1)
    ix=ieor(777755555,abs(idum))
    idum=abs(idum)+1
  endif
  ix=ieor(ix,ishft(ix,13))
  ix=ieor(ix,ishft(ix,-17))
  ix=ieor(ix,ishft(ix,5))
  k=iy/IQ
  iy=IA*(iy-k*IQ)-IR*k
  if (iy < 0 ) iy=iy+IM
  ran=am*ior(iand(IM,ieor(ix,iy)),1)
end subroutine

!=====

subroutine rand2(harvest)

  implicit none
  real :: v1,v2,ran2,rsq,harvest,g,harvest2
  integer idum,i

do
  call rand(idum,v1)
  call rand(idum,v2)
  v1=2.0*v1-1.0
  v2=2.0*v2-1.0
  rsq=v1**2+v2**2
  if (rsq > 0.0 .and. rsq < 1.0) exit
enddo

rsq=sqrt(-2.0*log(rsq)/rsq)
harvest=v1*rsq
g=v2*rsq
harvest=harvest*0.05
end subroutine

```

localsearch.f

```

subroutine localsearch(fit,pop,param,parlim,nparam,fx,m,cont,
&Verybest)

  implicit none
  integer i,j,k,m,n,p,pop,nparam,cont,ft
  double precision fit(pop,nparam+1,cont),Verybest(nparam+1),VAR
&,param(nparam,cont),parlim(nparam,2),incr,fax,value,stock,fx
&,repeat,cuim,efficient(nparam),sumeff,stock2,moyeff,d1,d2

  open(18,file='C:\buffer.dat',status='unknown')
  write(98,*) ''
  write(98,*) 'Local search'
  write(98,*) ''

  m=1
  d1=0.0
  d2=0.0
  cuim=0.0
  VAR=1000.0

  DO k=1,1000
    stock=Verybest(1)
    ft=mod(k,4)

    do i=1,nparam
      if (k.GT.1.AND.ft.NE.0.AND.efficient(i).LT.moyeff) then
        goto 38

```

```

endif
stock2=Verybest(1)

    if (VAR.GT.(0.5)) then
        repeat=0.00
    value=Verybest(1+i)
    do j=1,5
        incr=(parlim(i,2)-parlim(i,1))*0.125
    Verybest(1+i)=value+(real(j)-3)*incr
    if (Verybest(1+i).LT.parlim(i,1)) then
        Verybest(1+i)=parlim(i,1)
        repeat=repeat+1.0
    endif
    if (Verybest(1+i).GT.parlim(i,2)) then
        Verybest(1+i)=parlim(i,2)
        repeat=repeat+1.0
    endif
        do p=1,nparam
        param(p,1)=Verybest(1+p)
        enddo
        if (repeat.LT.(1.5)) then
    call func(param,nparam,fx,m,cont)
    else
        repeat=1.0
        endif
    write (18,*) fx, Verybest(1+i)
    enddo
rewind(18)
    fx=1000.0
    do j=1,5
        read (18,*) fax, value
    if (fax.LT.fx) then
        fx=fax
        Verybest(1+i)=value
        Verybest(1)=fx
    endif
    enddo
    value=Verybest(1+i)
    rewind(18)
else
    value=Verybest(1+i)
endif

if (VAR.GT.(0.01)) then
    repeat=0.00
    do j=1,5
        incr=(parlim(i,2)-parlim(i,1))*0.03125
    Verybest(1+i)=value+(real(j)-3)*incr
    if (Verybest(1+i).LT.parlim(i,1)) then
        Verybest(1+i)=parlim(i,1)
        repeat=repeat+1.0
    endif
    if (Verybest(1+i).GT.parlim(i,2)) then
        Verybest(1+i)=parlim(i,2)
        repeat=repeat+1.0
    endif
        do p=1,nparam
        param(p,1)=Verybest(1+p)
        enddo
        if (repeat.LT.(1.5)) then
    call func(param,nparam,fx,m,cont)
    else
        repeat=1.0
        endif
    write (18,*) fx, Verybest(1+i)
    enddo
rewind(18)
    fx=100000000000.0
    do j=1,5
        read (18,*) fax, value
    if (fax.LT.fx) then

```

```

        fx=fxax
        Verybest(1+i)=value
        Verybest(1)=fx
    endif
enddo
value=Verybest(1+i)
rewind(18)
endif

if (VAR.LT.(5.0).AND.VAR.GT.(0.1)) then
    repeat=0.00
    do j=1,5
        incr=(parlim(i,2)-parlim(i,1))*0.007813
        Verybest(1+i)=value+(real(j)-3)*incr
        if (Verybest(1+i).LT.parlim(i,1)) then
            Verybest(1+i)=parlim(i,1)
            repeat=repeat+1.0
        endif
        if (Verybest(1+i).GT.parlim(i,2)) then
            Verybest(1+i)=parlim(i,2)
            repeat=repeat+1.0
        endif
        do p=1,nparam
            param(p,1)=Verybest(1+p)
        enddo
        if (repeat.LT.(1.5)) then
            call func(param,nparam,fx,m,cont)
        else
            repeat=1.0
        endif
        write (18,*) fx, Verybest(1+i)
    enddo
    rewind(18)
    fx=100000000000.0
    do j=1,5
        read (18,*) fax, value
        if (fax.LT.fx) then
            fx=fax
            Verybest(1+i)=value
            Verybest(1)=fx
        endif
    enddo
    value=Verybest(1+i)
    rewind(18)
endif

if (d1.GT.(0.5)) then
    repeat=0.00
    do j=1,5
        incr=(parlim(i,2)-parlim(i,1))*0.001953
        Verybest(1+i)=value+(real(j)-3)*incr
        if (Verybest(1+i).LT.parlim(i,1)) then
            Verybest(1+i)=parlim(i,1)
            repeat=repeat+1.0
        endif
        if (Verybest(1+i).GT.parlim(i,2)) then
            Verybest(1+i)=parlim(i,2)
            repeat=repeat+1.0
        endif
        do p=1,nparam
            param(p,1)=Verybest(1+p)
        enddo
        if (repeat.LT.(1.5)) then
            call func(param,nparam,fx,m,cont)
        else
            repeat=1.0
        endif
        write (18,*) fx, Verybest(1+i)
    enddo
    rewind(18)
    fx=100000000000.0

```

```

do j=1,5
  read (18,*) fax, value
  if (fax.LT.fx) then
    fx=fax
    Verybest(1+i)=value
    Verybest(1)=fx
  endif
enddo
value=Verybest(1+i)
rewind(18)
endif

if (d2.GT.(0.5)) then
  repeat=0.00
  do j=1,5
    incr=(parlim(i,2)-parlim(i,1))*0.0004883
    Verybest(1+i)=value+(real(j)-3)*incr
    if (Verybest(1+i).LT.parlim(i,1)) then
      Verybest(1+i)=parlim(i,1)
      repeat=repeat+1.0
    endif
    if (Verybest(1+i).GT.parlim(i,2)) then
      Verybest(1+i)=parlim(i,2)
      repeat=repeat+1.0
    endif
    do p=1,nparam
      param(p,1)=Verybest(1+p)
    enddo
    if (repeat.LT.(1.5)) then
      call func(param,nparam,fx,m,cont)
    else
      repeat=1.0
    endif
  enddo
  write (18,*) fx, Verybest(1+i)
  rewind(18)
  fx=100000000000.0
  do j=1,5
    read (18,*) fax, value
    if (fax.LT.fx) then
      fx=fax
      Verybest(1+i)=value
      Verybest(1)=fx
    endif
  enddo
  value=Verybest(1+i)
  rewind(18)
endif

if (d2.GT.(0.5)) then
  repeat=0.00
  do j=1,5
    incr=(parlim(i,2)-parlim(i,1))*0.0001221
    Verybest(1+i)=value+(real(j)-3)*incr
    if (Verybest(1+i).LT.parlim(i,1)) then
      Verybest(1+i)=parlim(i,1)
      repeat=repeat+1.0
    endif
    if (Verybest(1+i).GT.parlim(i,2)) then
      Verybest(1+i)=parlim(i,2)
      repeat=repeat+1.0
    endif
    do p=1,nparam
      param(p,1)=Verybest(1+p)
    enddo
    if (repeat.LT.(1.5)) then
      call func(param,nparam,fx,m,cont)
    else
      repeat=1.0
    endif
  enddo
  write (18,*) fx, Verybest(1+i)

```

```

        enddo
    rewind(18)
    fx=1000000000000.0
    do j=1,5
        read (18,*) fax, value
    if (fax.LT.fx) then
        fx=fax
        Verybest(1+i)=value
        Verybest(1)=fx
    endif
    enddo
    value=Verybest(1+i)
    rewind(18)
endif

if (d2.GT.(0.5)) then
    repeat=0.00
    do j=1,5
        incr=(parlim(i,2)-parlim(i,1))*0.0000305
        Verybest(1+i)=value+(real(j)-3)*incr
    if (Verybest(1+i).LT.parlim(i,1)) then
        Verybest(1+i)=parlim(i,1)
        repeat=repeat+1.0
    endif
    if (Verybest(1+i).GT.parlim(i,2)) then
        Verybest(1+i)=parlim(i,2)
        repeat=repeat+1.0
    endif
    do p=1,nparam
        param(p,1)=Verybest(1+p)
    enddo
    if (repeat.LT.(1.5)) then
    call func(param,nparam,fx,m,cont)
    else
        repeat=1.0
    endif
    write (18,*) fx, Verybest(1+i)
    enddo
    rewind(18)
    fx=1000000000000.0
    do j=1,5
        read (18,*) fax, value
    if (fax.LT.fx) then
        fx=fax
        Verybest(1+i)=value
        Verybest(1)=fx
    endif
    enddo
    rewind(18)
endif
efficient(i)=ABS(stock2-Verybest(1))/(0.0001*stock2)
38 continue
    enddo
    sumeff=0.0
    do i=1,nparam
        sumeff=sumeff+ efficient(i)
    enddo
    moyeff=sumeff/real(nparam)
    VAR=ABS(stock-Verybest(1))/(0.010*stock)
write(98,*) ''
write(98,*) '-----'
write(98,*) ''
write(98,*) 'Local search round',k
write(98,*) 'Verybest=',Verybest(1)
write(98,*) 'Verybest/OldVerybest=',VAR,'% '
    if (VAR.LT.(0.5)) then
        d1=1.0
    endif
    if (VAR.LT.(0.1)) then
        d2=1.0
    endif

```

```

        if (VAR.LT.(0.0001)) then
            cuim=cuim+1
        else
            cuim=0.0
        endif
        if (cuim.GT.(4.5)) then
            goto 55
        endif

    ENDDO

55 continue
    write(98,*) ' '
    write(98,*) '-----'
    close(18)
end subroutine

```

output.f

```

subroutine output(gen,maxgen,generation,nparam,pop,
&parlim,fit,m,cont)

implicit none
integer i,j,k,m,II,pop,nparam,gen,maxgen,cont
double precision fit(pop,nparam+1,cont),sum,parlim(nparam,2)
&generation(pop,nparam,cont),fitnessmoyen(cont)
character *1000 lignes(1)

do k=1,cont
    fitnessmoyen(k)=0.0
    do i=1,pop
        fitnessmoyen(k)=fitnessmoyen(k)+fit(i,1,k)
    enddo
    fitnessmoyen(k)=fitnessmoyen(k)/pop
enddo

write(96,'(a,$)') 'Generation ';write(96,'(i5,$)') gen
write(96,'(a,$)') ' Continent ';write(96,'(i2)') m
write(96,*) ' ';write(96,*) ' - OF - '

do i=1,pop
    write(96,'(i4,$)') i
do k=1,nparam+1
    write(96,'(f14.8,$)') fit(i,k,m)
enddo
write(96,*) ' '
enddo

write(96,*) ' '
write(96,*) 'Mean fitness    =',fitnessmoyen(m)
write(96,*) 'Median fitness    =',fit(int(pop/2),1,m)
write(96,*) ' '
write(96,*) '-----'
&write(96,*) ' '
if (m.EQ.cont) then
    write(98,'(i6,$)') gen
    do i=1,cont
        write(98,'(f10.5,$)') fit(1,1,i)
    enddo
    write(98,*) ' '
endif

end subroutine

```

func.f

```

subroutine func(param,nparam,fx,m,cont)

```

```

    implicit none
    integer i,j,k,m,nparam,cont,par,II,depth,Mat,Lay,tAtm,rRoot,rB,
&T20,ADS,compt,NTab
    double precision tr,tm,hs,a,n,mm,l,Teta(2000),ks,ha,Kha,Se,elle,
&ts,FSe,fx,watercontent(20),pot(20),param(nparam,cont),fx,p(10)
&bestfo,xx,h,Beta,Axz,Bxz,Dxz,Prec,rSoil,hCritA,hB,ht,T1,T2,T3,T4
&T5,T6,T7,T8,T9,T10,T11(44),T12,T13,T14,T15,T16,T17,T18,T19,T21,
&T22,OBS(44),WEIGHT(44),saveTeta(100),savefx,savefxx,teta_i,h_i,
&erreurABS(44),erreurWAT(11)

    character *200 lig(1);character *82 chara82
    character *79 chara79;character *300 lignes(3000)
    character *300 selector(200)

ADS=11
hs=2.0

!Defining parameters =====

        tr=0.000
        ts=0.1849
        tr=param(1,m)
        a=param(2,m)
        n=param(3,m)
        mm=param(4,m)
        ks=10**param(5,m)
        elle=param(6,m)
        fx=0.000

!Evaluation of the retention curve =====

    open(20,file='C:\ABSORPTION\watercontent.DAT',status='old')
    read (20, '(A)',IOSTAT=II) lignes(1)
    do i=1,ADS
    read(20,*) pot(i),watercontent(i),WEIGHT(i)
        pot(i)=pot(i)*100.0
        tm=tr+(ts-tr)*(1+(abs(a*hs))**n)**mm
        Teta(i)=tr+(tm-tr)/(1+(abs(a*pot(i))**n)**mm

!calculating residual (OF)
        saveTeta(i)=Teta(i)
        erreurWAT(i)=WEIGHT(i)*(watercontent(i) - Teta(i))**2
        fx=fx+erreurWAT(i)
    enddo
    close(20)

    open(69,file='C:\ABSORPTION\Concrete\SELECTOR.IN',status='old')

!Evaluation of the hydraulic conductivity curve =====

    open(70,file='C:\ABSORPTION\Concrete\Mater.in',status='unknown')
    open(71,file='C:\ABSORPTION\Fs\prep-Mater.in',status='old')
    read(71,*) NTab
    write(70,*) 'icap'
    write(70,*) '0'
    write(70,*) 'NTab'
    write(70,'(i3)') NTab
    write(70,*) 'theta          h          K'
    do i=1,NTab
    read(71,*) ha
    if (ha.lt.hs) then
        Teta(i)=ts
        Kha=ks
    else
        tm=tr+(ts-tr)*(1+(abs(a*hs))**n)**mm
        Teta(i)=tr+(tm-tr)/(1+(abs(a*ha))**n)**mm
        Se= (Teta(i)-tr)/(tm-tr)
        FSe=(1.0-Se**(1.0/mm))**mm
        Kha=ks*(Se**elle)*(1-(1-Se**(1.0/mm))**mm)**2.0
    endif
    write(70,'(f8.5,e15.4,x,e14.4)') Teta(i),-ha,Kha

```

```

        enddo

        close(71)

!Modifying file Selector.in =====

        do i=1,46
        read (69, '(A)',IOSTAT=II) selector(i)
        if (II.NE.0) exit
        end do
        rewind (69)
        do i=1,28
        write(69,'(A)') TRIM (selector(i))
        end do

        write (69,125) tr,ts,ks

        do i=30,46
        write(69,'(A)') TRIM (selector(i))
        end do

        close(69)

!Modifying file PROFILE.DAT =====

!saturation degree
        Se=0.9031

!initial water content and initial pressure head
        teta_i=Se*(tm-tr)-tr
        h_i= (((tm-tr)/(teta_i-tr))**(1.0/mm)-1)**(1.0/n))/(-1.0*a)

        open(60,file='C:\ABSORPTION\Concrete\PROFILE.DAT',status='old')
        open(99,file='C:\ABSORPTION\Concrete\buffer.DAT',status='new')

        do i=1,5
        read (60, '(A)') lignes(i)
        write(99,'(A)') TRIM (lignes(i))
        end do

        do i=1,200
        read (60,*,IOSTAT=II) depth,xx,h,Mat,Lay,Beta,Axz,Bxz,Dxz
        write(99,780) depth,xx,h_i,Mat,Lay,Beta,Axz,Bxz,Dxz
        end do

        depth=201

!boundary conditions
        xx=-5.0
        h=0.500
        write(99,780) depth,xx,h,Mat,Lay,Beta,Axz,Bxz,Dxz
        write(99,*) ' 0'

        close(60)
        close(99)
        call system('MOVE C:\ABSORPTION\Concrete\buffer.DAT C:
&\ABSORPTION\Concrete\PROFILE.DAT' )

!Running HYDRUS-1D =====

        call system('C:\ABSORPTION\run.bat' )

!Computing the objective function =====

        fxx=0.0
        rewind(13)
        open(51,file='C:\ABSORPTION\Concrete\T_LEVEL.OUT',status='old')
        do i=1,9

```

```

read (51, '(A)', IOSTAT=II) lig(1)
enddo
do i=1,44
  read(51,*, IOSTAT=II) T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11(i)
  T12,T13,T14,T15,T16,T17,T18,T19,T20,T21,T22
  if (II.NE.0) then
    write (*,*) 'PROBLEME LECTURE DATA DANS T_LEVEL.OUT'
    fxx=fxx+100.0
  endif
  read (13,*, IOSTAT=II) OBS(i),WEIGHT(i)
  if (II.NE.0) then
    write(*,*) 'IOSTAT=',II
    write(*,*) i
    write(*,*) OBS(i),WEIGHT(i)
    write (*,*) 'PROBLEME LECTURE DATA DANS observation.dat'
48    continue
    goto 48
  endif
  if (T11(i).EQ.0.0) T11(i)=0.000000000001
!calculating residual (OF)
  erreurABS(i)=WEIGHT(i) * (T11(i)-OBS(i)) **2
  fxx=fxx+erreurABS(i)
  enddo

  savefx=fx
  savefxx=fxx
  fx=fx+fxx
  close(51)

!Saving results if new-best OF found  =====

  rewind(93)
  read (93,*) bestfo

  if (bestfo.GT.fx) then
    rewind(93)
    write(93,110) fx
    write(93,*) ' thr ths a(cm-1) n m k
    &s(cm/min) l'
    write (93,125) tr,ts,a,n,mm,ks,elle
    rewind(92)
    write(92,*) 'OBSERVATION PREDICTION RMSE'
    do i=1,44
      write(92,200) OBS(i),T11(i),erreurABS(i)
    enddo
    write(92,*) ' '
    do i=1,ADS
      write(92,200) watercontent(i),saveTeta(i),erreurWAT(i)
    enddo
    write(92,*) ' '
    write(92,*) 'RMSE water content',savefx
    write(92,*) 'RMSE absorption ',savefxx
    rewind(70)

    call system('COPY C:\ABSORPTION\Concrete\Mater.in C:
    &\ABSORPTION\Concrete\BESTMater.in' )
    call system('COPY C:\ABSORPTION\Concrete\SELECTOR.IN C:
    &\ABSORPTION\Concrete\BESTSELECTOR.IN' )
    call system('COPY C:\ABSORPTION\Concrete\PROFILE.DAT C:
    &\ABSORPTION\Concrete\BESTPROFILE.DAT' )
    endif

110  FORMAT (e14.6)
125  FORMAT (2f8.4,3x,e10.4,3x,e10.4,f8.3,2x,e9.3,x,f7.3)
200  FORMAT (3f11.4 )
780  FORMAT (i5,2e15.6,2i5,4e15.6)

  close(70)
  return
end subroutine

```

config.dat

```
integer nparam, pop, maxgen, cont, mix, elitism, locals
real mfreq, mint
```

```
parameter( nparam=6 )
parameter( pop=200 )
parameter( maxgen=40 )
parameter( cont=1 )
parameter( mix=3 )
parameter( elitism=1 )
parameter( mfreq=0.20 )
parameter( mint=0.75 )
parameter( locals=1 )
```

parameter.inc

```
Parameter ranges
0.000      0.070
0.000001  0.00001
1.05      2.000
0.2500    0.500
-10.2218  -7.2218
-3.0      50.0
END
```

The previous values in the parameter.inc file correspond, respectively, to the lower and upper boundaries of the following parameters of the van Genuchten model:

- θ_r (residual water content, $\text{cm}^3 \text{cm}^{-3}$)
- α (curve shape parameter, m^{-1})
- n (curve shape parameter)
- m (curve shape parameter, $m = 1 - 1/n$)
- K_s (saturated hydraulic conductivity, $\log_{10}$ (m/s))
- l (Mualem parameter in relative hydraulic conductivity relationship)

4 Examples

Literature on global optimization test problems provides numerous complex mathematical functions, which own several local minimum, and thus can be considered as good global optimization test problems. Nevertheless those functions are far different from the non-linear set of equations pertaining to water flow problems in unsaturated media. We verified that the GENAPAC is able to solve correctly most of these problems (not shown here). In this section, we will provide examples pertaining to the field of water fluxes in porous media, in order to demonstrate the versatility of the GENAPAC code in calibrating complex non-linear hydraulic relationships. Therefore two example of hydraulic parameter optimization will be given, one for a concrete sample and one for a soil profile. A description of the problem, the objective function, the tuning of GENAPAC and the output files will be provided.

4.1 Calibration of unsaturated hydraulic parameters of concrete

Definition of the optimization problem

This example is based on the study of Schneider et al. (2010a). The latter study aims at estimating the hydraulic properties of specific concrete samples, by using water retention curve data combined with data from a capillary suction experiment (cumulative water uptake). The principle of the capillary suction experiment is described by Fagerlund (1986). Basically, concrete specimens have been equilibrated with an atmosphere having 54% of relative humidity. Then, the bottom part of the specimen has been put in a water reservoir with the water level reaching 0.005 m above the specimens' bottom surface, whereas the other surfaces of the specimens have been covered with plastic to make them impermeable and avoid water loss by evaporation. By measuring the sample weight at different times (three samples were used), the evolution of water absorbed by capillary suction was recorded. Experimental data are shown in Figure 2.

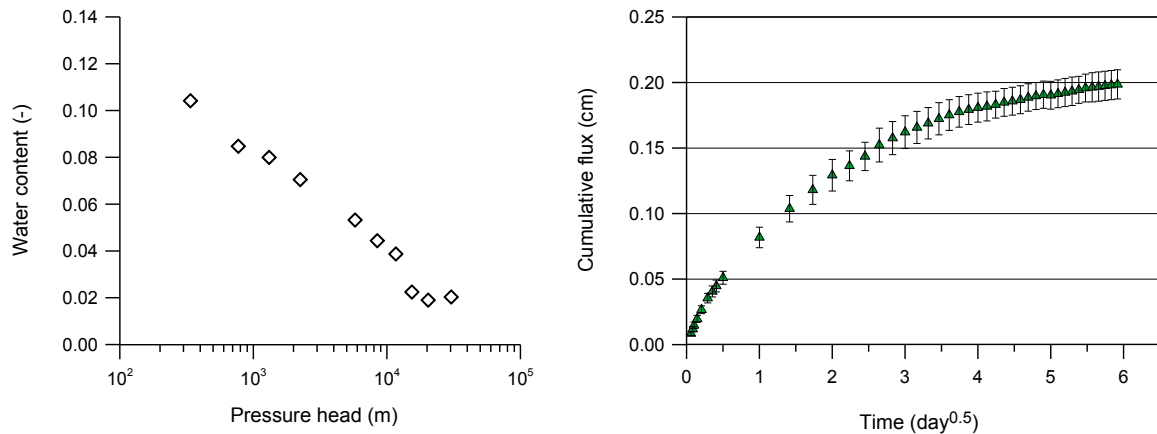


Figure 2. Water retention data (left) and cumulative flux across the concrete specimens averaged over three samples (vertical error bars represent one standard deviation) (right).

Using the HYDRUS-1D software package (*Šimůnek et al.*, 2009), a one-dimensional simulation of this experiment has been built. The main characteristics of the conceptual model can be summarized as follows:

- domain height: 0.05 m
- lower boundary condition: constant pressure head equal to +0.005 m. This boundary condition reflects the fact that the sample has been immersed in water by 5 mm.
- upper boundary condition: zero flux (specimen covered with plastic).
- use of a single porosity hydraulic model: van Genuchten - Mualem (VGM), with condition of *Schaap* (2006) which implements an air entry condition of $h_s = -2$ cm.
- a spatially uniform initial pressure head condition in the entire sample was considered.

Because two types of data are taken into account in this optimization process, the following formulation of the objective function, OF, was used:

$$OF = \sum_{i=1}^M w (q_{i^*} - q_i)^2 + \sum_{j=1}^N \frac{1}{w} (\theta_{j^*} - \theta_j)^2 \quad (2)$$

where M and N represent the number of measurements of cumulative flux and water retention data (i.e. water content) respectively, q_{i^*} and q_i are the i th measured and predicted cumulative flux, respectively, θ_{j^*} and θ_j are the j th measured and predicted water content, respectively, and w is a weighting factor introduced in order to give both data set a similar weight, which is defined as:

$$w = \frac{\sum_{j=1}^N \theta_{j^*}}{\sum_{i=1}^M q_{i^*}} \quad (3)$$

The file parameter.inc which tunes GENAPAC was the following:

```
integer nparam, pop, maxgen, cont, mix, elitism, locals
real mfreq, mint
```

```
parameter( nparam=6 )
parameter( pop=200 )
parameter( maxgen=40 )
parameter( cont=1 )
parameter( mix=10 )
parameter( elitism=1 )
parameter( mfreq=0.20 )
parameter( mint=0.75 )
parameter( locals=1 )
```

Results

The water retention curve (left) and the cumulative flux across the concrete (right) obtained with the optimized van Genuchten parameters using GENAPAC are shown in Figure 3.

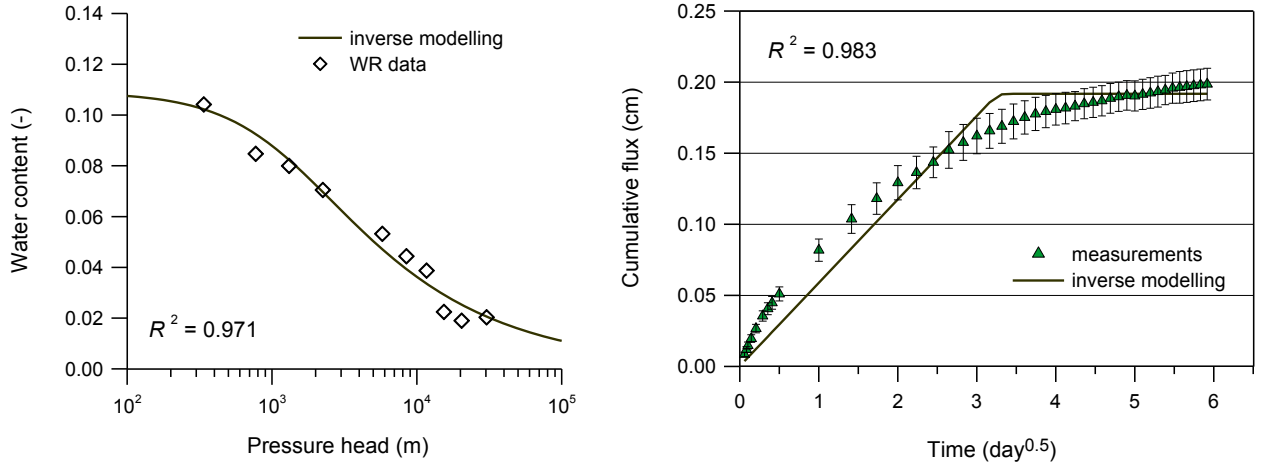


Figure 3. Water retention curve (left) and cumulative flux across the concrete (right) obtained with the optimized van Genuchten parameters using GENAPAC.

In the file STAT.OUT (not shown), one can see that the objective function (OF) started from a value of 36.5 at the end of the first generation, and decreased to 10.4 at the last generation. Afterwards, the local search started and succeeded to further decrease the OF value (equal to 9.52 at the end of the local search). When looking at the residual (file BESTresinv.out, not shown), it appears that the contribution of the two data sets to the OF are unequal: the prediction errors on water content data contribute to 15% of the OF value (OF value equal to 1.4) whereas the prediction errors on cumulative flux contribute to 85% of the OF value (OF value equal to 8.1). This is consistent with Figure 3 which shows a good fit for the water retention curve, whereas the fit of the cumulative flux across the concrete sample shows systematic errors (under-estimation in the first part of the curve, over-estimation in the middle part, and under-estimation in the last part). These systematic errors are likely to be due to the combination of experimental errors (i.e. preparation of the samples) and the inability of the hydraulic model to take into account complex phenomena that occur in fresh concrete submitted to wetting.

4.2 Calibration of unsaturated hydraulic parameters for a heterogeneous soil profile

Definition of the optimization problem

This example is based on the study of Schneider et al, (2010b). This study aims at calibrating hydraulic properties of the different layers of a heterogeneous sandy soil. The experiment took place in Belgium in the Campine region (Mol, SCK•CEN domain). The soil is classified as a podzol. A lysimeters has been installed for the purpose of soil water monitoring. The sequence of soil layers is: (i) a top layer (litter) of about 13-cm thick, (ii) a 37-cm thick eluvial horizon with anthropogenic disturbances consists of two sub-horizons (a light layer on top of a black organic-rich one), and (iii) the parent material with two sub-horizons: a white-colored sub-horizon, and a green-colored sub-horizon starting from a depth of 88 cm. Soil water contents were recorded using time domain reflectrometry (TDR) probes. Three-rod TDR probes 35-cm long were installed through the walls of a 40-cm diameter PVC cylinder. Roots were cut during the installation of the PVC cylinder to exclude effects of root water uptake on the soil-water balance. TDR probes were installed at the depths of 10, 20, 30, 40, 50, 60, 70 and 90 cm. Water contents were recorded daily using a Tektronix 1502C cable tester connected to a datalogger (CR10X, Campbell Inc.). Data are shown in Figure 4. Vegetation data, meteorological data, and

groundwater table data were also measured, and used for implementing the boundary conditions in HYDRUS-1D.

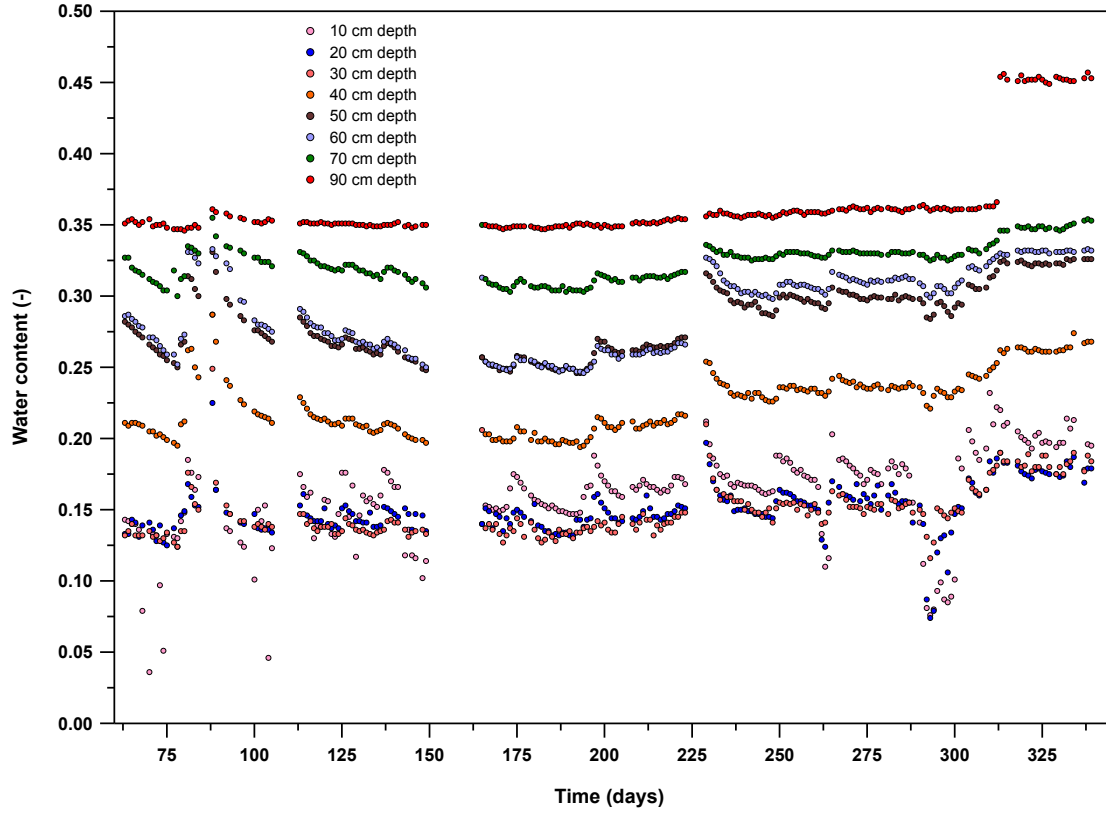


Figure 4. Volumetric water contents measured with TDR.

The following formulation of the objective function, OF , was used:

$$OF = \sum_{i=1}^N [\theta_i - \theta_i^*]^2 \quad (4)$$

where N is number of water content measurements and θ_i and θ_i^* are the i th observed water content and the i th calculated water content, respectively. The number of parameters which have been optimized is 25 (5 parameters per layer: 5x5).

The file parameter.inc which tunes GENAPAC was the following:

```
integer nparam, pop, maxgen, cont, mix, elitism, locals
real mfreq, mint

parameter( nparam=25 )
parameter( pop=200 )
parameter( maxgen=300 )
parameter( cont=5 )
```

```
parameter( mix=15      )  
parameter( elitism=1    )  
parameter( mfreq=0.20  )  
parameter( mint=0.75   )  
parameter( locals=1    )
```

Results

Analysis of file STAT.OUT (not shown) revealed that the objective function started from a value of 2.7 at the end of the first generation, and decreased to 0.48 at the last generation.

Simulated water contents at the different depths and times are plotted in **Figure 5**. Observed and simulated daily water contents at 10, 20, 30, 40, 50, 60, 70, and 90 cm depth. It appears that the fits are good in general (mean error inferior to $0.02 \text{ cm}^3 \text{ cm}^{-3}$), but some systematic errors (over or under-estimation) occur, for instance at 60 cm depth. These systematic errors are likely to be due to experimental errors, or the inability of the conceptual model implemented in HYDRUS-1D to perfectly mimic the soil water fluxes.

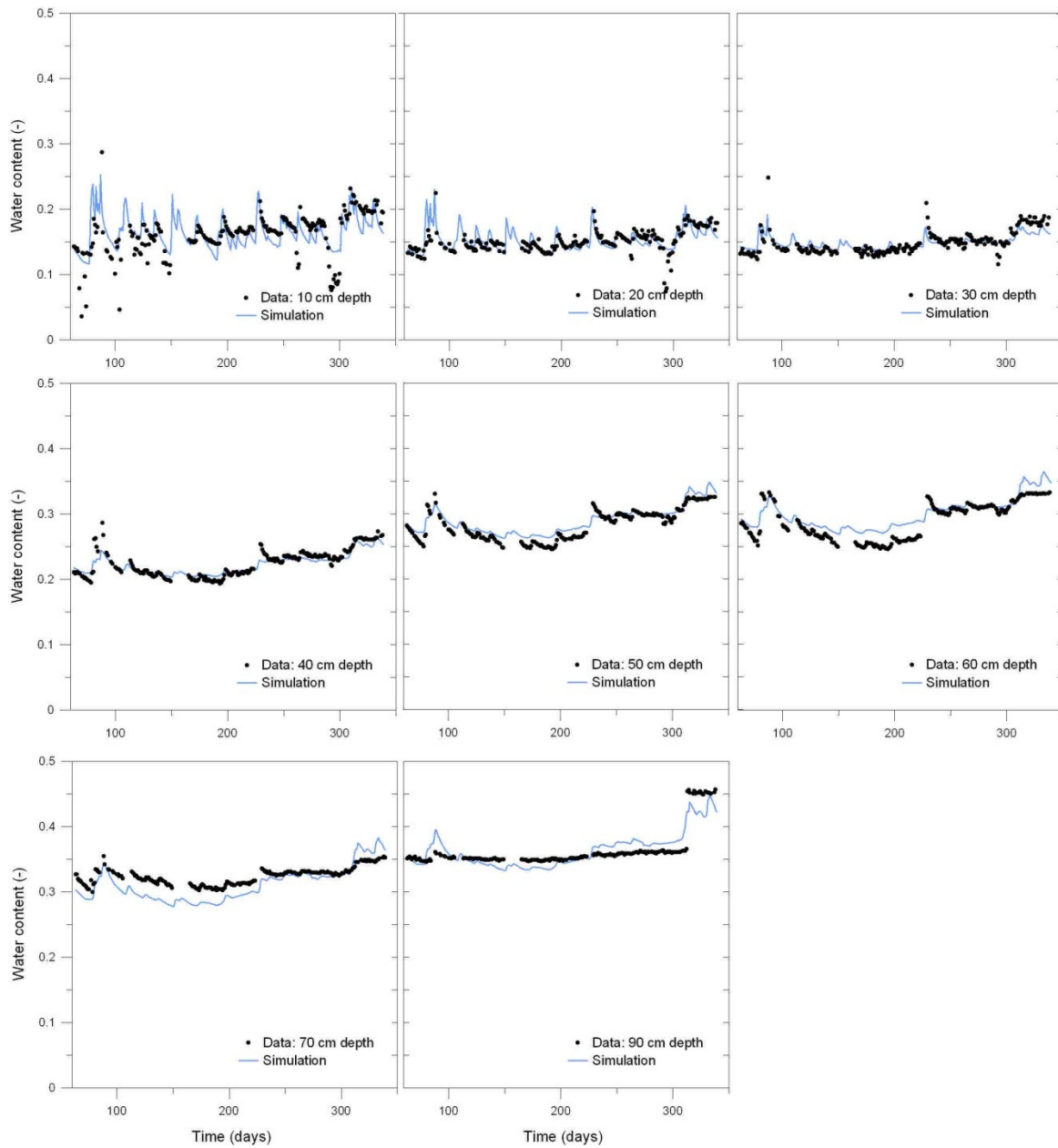


Figure 5. Observed and simulated daily water contents at 10, 20, 30, 40, 50, 60, 70, and 90 cm depth.

5 Conclusion

This user's manual provides information for using the computer code GENAPAC. After having given some general information about genetic algorithms, this manual has provided source code and additional input and output files, as well as two examples of a soil hydraulic parameter optimization problem. For running these problems, additional files which are not provided in this document, are compulsory. Those files are the HYDRUS-1D source files that can be obtained by downloading the free HYDRUS-1D package (<http://www.pc-progress.com>), and the HYDRUS-1D input files which can be built by following the description of the problem as documented in Schneider et al. (2010a) and Schneider et al. (2010b).

6 Acknowledgement

This work was supported by a research grant from SCK•CEN (AWM Postdoc) and NIRAS/ONDRAF.

7 References

- Fagerlund G., 1986. On the capillarity of concrete, Nordic Concrete Research, no.1, Oslo, Ppe No.6.
- Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.
- Goldberg, D.E., Deb, K., 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, in: Foundations of Genetic Algorithms, in: Rawlins, G.J.E., Kaufmann M. (Eds.), San Mateo, CA, pp. 69-93.
- Ines, A.V.M., Mohanty, B.P., 2008. Near-surface soil moisture assimilation for quantifying effective soil hydraulic properties under different hydroclimatic conditions. *Vadose Zone J.* 7, 39-52.
- Schneider, S., D. Jacques, D. Mallants, 2010a. Estimating unsaturated hydraulic properties of concrete C-15-A and mortar M1. Report SCK•CEN-R-5051, Mol, Belgium.
- Schneider, S., D. Jacques, and D. Mallants, 2010b. Modelling water fluxes in a pine wood soil-vegetation-atmosphere system. Comparison of a water budget and water flow model using different parameter data sources. SCK•CEN BLG-1071, 85 p.
- Šimůnek, J., M. Šejna, and M. Th. van Genuchten, 2009. The HYDRUS-1D software package for simulating the one-dimensional movement of water, heat, and multiple solutes in variably saturated media. Version 4.12, HYDRUS Software, Department of Environmental Sciences, University of California Riverside, Riverside, CA, 281 pp.
- Zhang, X., Srinivasan, R., Bosch, D., 2009. Calibration and uncertainty analysis of the SWAT model using Genetic Algorithms and Bayesian Model Averaging. *J. Hydrol.* 374, 307-317.

		Date	Approval
Author:	Sébastien Schneider	22.11.2010	P.P.  DIRK MALLANTS
Verified by:	Joan Govaerts	11.01.2011	
QA verification:	Elke Jacops	24/01/2011	
Approved by:	Dirk Mallants	7/02/2011	