MODELING SOME DYNAMIC PHENOMENA WITH MAPLE6 IN A CAS-BASED MATH CLASS

Patricia E. Balderas Cañas, Jacinto Mendez Banda, and
Xavier Rojel Martinez
Facultad de Ingeniería, Universidad Nacional Autónoma de México

Abstract

We are interdisciplinary members of a research team supported by CONACYT #41596-Y, Mexico. We pursuit some math teaching guidelines for modeling courses at a main Mexican University. At the Summer Academy, we will illustrate some classroom activities from CAS-based curricula and teaching methods approach. The modeling problem is get the valve diameter of a pipeline entrance as a result of a decision making process from the waste water flow that is poured into one tank of a waste water treatment process. We designed two Maple6 programs to generate a data table of the water flow historical records, the correspondent residence time and the valve diameter. The residence time is an intermediate variable and the valve diameter is the key decision. This is very important because is the control for sewage plant. We think that this is the type of problems that we should discuss in a CAS-based teaching of applied mathematics on transport phenomena.

Introduction

Perhaps, many of us as mathematics teachers have listened that some curriculum boards try to increase the number and the extent of math courses of engineering curricula, and from professional engineering activities, some engineers judge that math curricula is huge. So, curriculum boards should find a balance between those points of views.

In addition, there are still some methodological debates related with an extensive use of specific software for math courses. Both problems aloud us to look for some math teaching guidelines, particularly with modeling courses for engineering curricula at The National Autonomous University of Mexico.

Indeed, undergraduate teaching has a huge challenge when its main objective is promote student's conceptual change and math professor is the promoting agent of that conceptual change. Modeling is the mean by which this teaching would reach that objective.

Curricula and teaching contexts

We analyzed curricula from three points of view: thematic content, pertinence and coherence. First, did thematic content reflect the relation between professional activities and disciplines? Second, was engineer profile pertinent to professional requirements? And third, was math modeling background of math professors coherent with resources (technology and material) available to institution? Those questions address our research activities. Somewhere, we report some coincidences, repetitions and lacks of modeling topics in mathematics curricula at undergraduate and graduate levels (Balderas, *et.al.* 2003).

Simultaneously, we did analyses of postgraduate teaching and found that the teaching trends were based on case discussions; most of them came from professional experience. In those cases, we remark the priority to systematize that teaching at the time teacher's research on the same area.

Following the third point of view, we designed some activities based on two Maple6 programs (see the Annex) that produce historical and numerical data from hydraulic flow inputs. The modeling problems were to get residence times and valve diameters (a cutoff valve) of a pipeline entrance as a result of decision making processes from the waste water flow that is poured into one tank of a waste water treatment process. Residence times and valve diameters (of the outlet size) were the outputs. The residence time is an intermediate variable and the valve diameter is the key decision. This is very important because is the control for sewage plant.

Model building

Our team visited three wastewater treatment plants. Those plants were built as part of a huge project of Puebla State Government. The treatment process is represented in figure 1. It depicts the main steps of treatment process of wastewater that were our focus. A first step is collect wastewater (1600 cubic meters). Second, after the water is pumped at 1.2 cubic meters by second, a fine sifting eliminates solid waste and feeds a deposit where grease and oil are removed by a mechanical device. At the same time, sand is removed by decantation.

Then, the organic material is separated from water by a flocculation process that produces sludge. Next, the sludge is sent to an anaerobic digestion process, and a chlorination process treats the water, free of organic material. After this last process, a final product is gotten that we name treated water.

The treatment process is conceived as an open and continuous system, and it is graphically represented in Figure 1. The numbered processes from (1) to (3), as black boxes, were modeled with the rule *input – treatment – output*. The first Maple6 program, for residence times (see the

Annex), corresponds to the process (1) as a function of flow. The second program shows draining times vs. outlet sizes, in process (2).

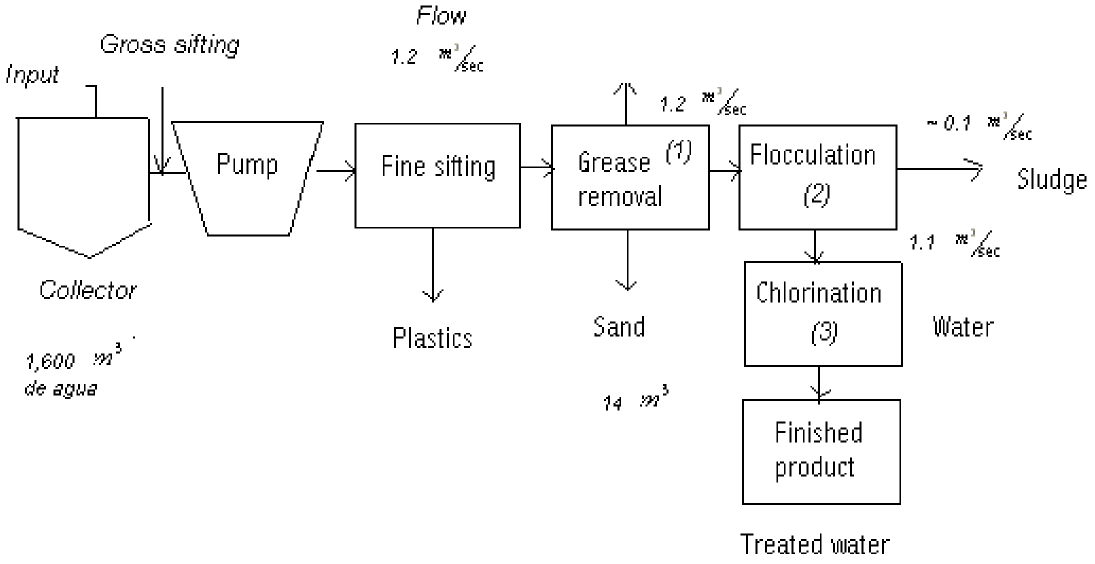


Figure 1. An open and continuous system of sewage treatment

In order to build a model we used a schema (see Figure 2) to represent the computational mechanics[1]. Many physical problems of mechanical engineering, related with solid body, mechanics of flow, stiffness body, etc., follow the principles of Physics of continuum media (Gurtin, 1981; Mase & Mase, 1999). We assume the following continuous media variation principia.

1. Energy conservation
2. Mass conservation
3. Linear momentum conservation
4. Entropy

---

[1] Méndez-Banda, Jacinto R. (2004) Metodología para el planteamiento y análisis de problemas de corrosión de acero en concreto bajo atmósfera marina. Posgrado de Ingeniería Mecánica, Facultad de Ingeniería, Universidad Nacional Autónoma de México.

The key question is how long it takes to drain water from a tank? The tank is part of the wastewater treatment system. We set the following hypothesis to explain the water flow behavior. Water follows Newton laws, so our framework is the mechanics of continuous media. But, there were some restrictions: atmosphere pressure, laminar flow, hydraulic time, waste water comes from municipal zones, and the treatment process is an advanced primary one.

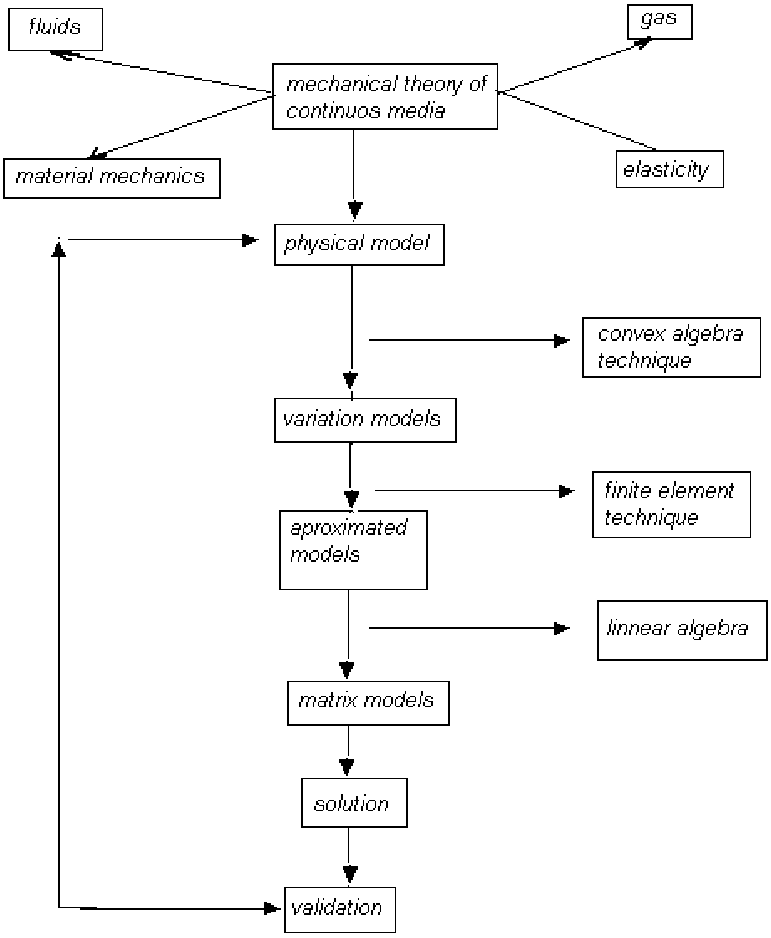


Figure 2. Schema of a computational mechanics

As the water is stored it has potential energy [1], where variables are m = mass, g = gravity constant, and h = height. If a drop falls free, its kinetic energy is $\frac{1}{2}mv^2$ [2]. So, by the energy conservation principle we equate [1] and [2].

$$\text{Kinetic energy} = \text{potential energy} \qquad [3]$$

$$\frac{1}{2}mv^2 = mgh \qquad [4]$$

Then, we solve for v

$$v = \sqrt{2gh} \qquad [5]$$

From that, we state a hydrodynamic expression known as the Torricelli's Law as follows. Let A_t be the cross draining area, so the water flow that comes out from a tank is

$$\text{Output flow} = A_t\sqrt{2gh} \qquad [6]$$

Then, the corresponding differential equation is

$$\frac{dV}{dt} = -A_t\sqrt{2gh} \qquad [7]$$

We are able to predict hydraulic time as we increase or decrease draining area; because, grease and oil removing takes time.

Generally speaking, for a primary advanced process we consider four tanks: one collector, one grease and oil removing tank, one flocculation tank, and one chlorination tank. With a properly modification of equation [6] we model each tank of the system.

The set of equations and operating conditions produce a model at each step, from hydraulic point of view, when $g = 9.81 \ {}^{m}\!/_{\sec^2}$, $h = 4 \ m$ and 0.8 is the friction constant.

Concept	Model
Grease removal	$GR = 0.8A_T\sqrt{(2)(9.81)(4)}$
Flocculation	$F = 0.8A_T\sqrt{(2)(9.81)(3)}$
Chlorination	$C = 0.8A_T\sqrt{(2)(9.81)(2)}$

Table 1. Math modeling of some physical processes. Concepts and formulations.

Class methodology

At the beginning of an applied math course, students form small groups (with three or four members). Those groups propose some ideas to develop projects that should be negotiated with the full class and professor. Then, as the course goes, thematic discussions are oriented to reach most of project objectives. In that way, students are deeply engage with their own project and the class progress.

An introduction to Maple6 environment and commands is necessary if students do not know them. At this part, students frequently propose the use of other software resources, like Matlab, Statistics or Excel, to develop their projects. So, this challenges professor's management class because, she must coordinate and focus class activities to reach the objectives of applied math course.

Now, let us depict a sequence of activities that we propose to develop in applied math class sessions, repeatedly.

Classroom activities

1. Set the objective(s) of modeling a system (as an example, the wastewater treatment plant).
2. Review and analyze pertinent literature to the system (mechanics of continuous media, in this case).
3. Select software resource(s) to build a model (Maple6).
4. Collect data (historical data from waste water treatment plant).
5. Validate and test the model (use pertinent methods to do that).
6. Make conclusions and decisions.

Some remarks

The interpretative context of our didactical proposal was professional engineer work. So, from teaching analysis and modeling activities, we suggest that mathematics curricula at engineer schools should be reviewed to inquiry on previous three questions (see curricula and teaching section).

As our discussions took place, engineer, biologist, mathematician, and mathematics educator, negotiated basic meanings to reach a gradual comprehension of wastewater treatment process. Gradual formulations of the treatment process incorporated those negotiated meanings (from arithmetic to higher mathematics). So, we agree with Jorgensen (2003, 880) that said "...[i]ndependent lines of thinking fortuitously converge on a common ground of some algorithms and some matrix tricks,..., the same mathematical term being assigned different names by the different groups, and yet the discussion is about the same fundamental underlying idea…"

Finally, we think that professional activities are the source and they should inspire us to set proper problems to discuss in a CAS-based teaching of applied mathematics.

References

Balderas-Cañas, P., Flores de la Mota, I. , and Nivón-Zaghi, A. *Análisis curricular de matemáticas en la licenciatura y maestría en ingeniería respecto a la modelación de fenómenos dinámicos.* XXXVI Congreso Nacional de la Sociedad Matemática Mexicana, Pachuca, Hgo. México, October 2003.
Gurtin, M.E. (1981) *An introduction to Continuum Mechanics*. London: Academic Press, Inc.
Jorgensen, P. E. T. (2003) "Matrix Factorizations, Algorithms, Wavelets". *Notices of the American Mathematical Society*, 50 (8), September.
Mase, G. T. and Mase, G.E. (1999) *Continuum Mechanics for Engineers*. Boca: CRC Press.

Annex

Program 1. Residence Time

```
> #Project 41596-Y SEP-CONACYT
Researcher on Chief: Patricia E Balderas Cañas
Participants: Jacinto R Mendez Banda and Xavier Rojel Martinez
Engineering School, The National Autonomous University of Mexico

> residence_time:=proc(file,data_num::numeric, nonnegint)
>  #Global variables definition
  #Local variables definition
>      global f, time_hrs, time_min:
   local  contador, data_array1, data_array2, v, i, c:
>   data_array1:=array(1..data_num):
   data_array2:=array(1..data_num):
>   v:=Vector(1..data_num):
>  #Importation of data
>   v:=ImportVector("a:\\octnovdicdata.txt",format=rectangular);
>  #Data processing
>    for contador from 1 to data_num do:
>    data_array2[contador]:=v[contador]:
>  end do:
>  #Calculation of residence time
>       with(linalg):
    for i from 1 to data_num do:
> f:=(i)->204/((0.002026*95)*0.8*sqrt((2*9.81*(v[i]))/(4*17*24*60*60)));
  end do:
   time_hrs:=Vector(data_num,f):
>      time_min:=scalarmul(time_hrs,1/60);  print(time_min);
>  #Data plotting
>   with(plots):
   pointplot({seq([i,time_min[i]],i=1..data_num)}):
> end proc;
```

Compilation

```
residence_time := proc (file, data_num::numeric, nonnegint)
local  contador, data_array1, data_array2, v, i, c;
global f, time_hrs, time_min;
       data_array1 := array( 1 .. data_num );
       data_array2 := array( 1 .. data_num );
```

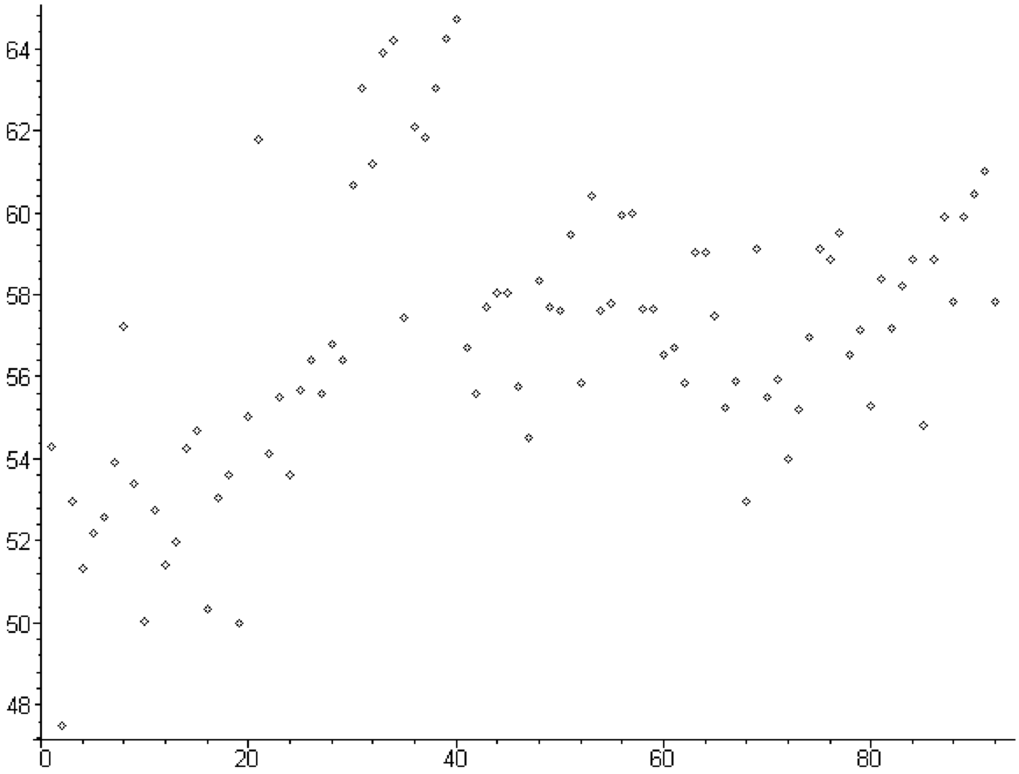
```
     v := Vector( 1 .. data_num );
     v := ImportVector( "a:\\octnovdicdata.txt", format = rectangular );
     for contador to data_num do data_array2[ contador ] := v[ contador ] end do ;
     with( linalg );
     for i to data_num do f := i → 1324.881800×1/sqrt( .3339460784*10^(-5)×v[ i ] )
     end do ;
     time_hrs := Vector( data_num, f );
     time_min := scalarmul( time_hrs, 1/60 );
     print( time_min );
     with( plots );
     pointplot( { seq( [ i, time_min[ i ] ], i = 1 .. data_num ) } )
 end proc
```

Program Execution

```
> residence_time("a:\\octnovdicdata.txt",92);
Warning, the protected names norm and trace have been redefined and unprotected

 [ 54.29421345, 47.47891233, 52.97375967, 51.32338243, 52.20190540, 52.55622105,
     53.89311262, 57.24204212, 53.38056290, 50.05704412, 52.75620738, 51.43485417,
     51.97444503, 54.27230293, 54.66520468, 50.32549487, 53.04517353, 53.61124018,
     49.97990797, 55.01537283, 61.78342643, 54.12509765, 55.49485643, 53.61651767,
     55.69492337, 56.40029390, 55.57109775, 56.78511383, 56.42488527, 60.67521935,
     63.01455810, 61.17078832, 63.88931618, 64.20419888, 57.44220102, 62.10083005,
     61.81575685, 63.04886107, 64.24956328, 64.72712522, 56.69129027, 55.58873658,
     57.69044548, 58.04211863, 58.04211863, 55.75417963, 54.52033690, 58.33901122,
     57.69044548, 57.59207032, 59.46552650, 55.86728947, 60.39415578, 57.62480610,
     57.77611302, 59.94656445, 59.96870813, 57.67401462, 57.67401462, 56.52975858,
     56.71001778, 55.83745702, 59.03107587, 59.03812142, 57.47468173, 55.25066625,
     55.89118993, 52.95340865, 59.11578922, 55.50656545, 55.93908298, 54.00603423,
     55.18148758, 56.96150468, 59.12286517, 58.84179457, 59.49435128, 56.54832643,
     57.13954957, 55.30272133, 58.38666545, 57.19072697, 58.21024977, 58.87670930,
     54.81123343, 58.87670930, 59.88763443, 57.81578078, 59.87292907, 60.45459528,
     60.99906953, 57.84227142 ]

Warning, the name changecoords has been redefined
```



Program 2. Draining time vs outlet size

```
> #Project 41596-Y SEP-CONACYT
Researcher on Chief: Patricia E Balderas Cañas
Participants: Jacinto R Mendez Banda and Xavier Rojel Martinez
Engineering School, The National Autonomous University of Mexico

> #Program: DRAINING TIME VS OUTLET SIZE
>
> Outlet_size:=proc(file,data_num,data_view,intervals::numeric, nonnegint)
>
> # The procedure has the following parameters
> # Parameter 1 <<file>> Data are taken form the file
> # Parameter 2 <<data_num>> The amount of data in the file that would be processed
> # Parameter 3 <<data_view>> Ask if before data processing it should be display them
> # Parameter 4 <<intervals>> Amount of intervals to associate outlet sizes
> # Variables definition
>
>    global f, dt, v, os,
>         draining_time,
>         draining_time_Min,
>         draining_time_Max,
>         upper_limit,
>         lower_limit,
>         result,
>         points;
```

```
>
>   local chain,
>        imported_data,
>        flow,
>        counter, aux_counter,
>        m, Ku, M, K,
>        k,
>        finding;
>
>   imported_data    := Vector(1..data_num):
>   flow             := Array(1..data_num):
>   draining_time    := Array(1..data_num):
>   lower_limit      := Array(1..intervals):
>   upper_limit      := Array(1..intervals):
>   result           := Array(1..data_num):
>   dt               := Array(1..data_num):
>   v                := Vector(1..data_num):
>
>   chain := `File from we take  `:
>   chain := cat(chain,data_num,` data:  ->  `):
>   chain := cat(chain,file):
>   print(chain) :
>
> # (1) Data importation
>   imported_data:=ImportVector(file,format=rectangular);
>
> # (2)Data transformation as an array to process them
>    flow:= imported_data:
>
> # (3) Charging libraries
>   with(linalg):
>   with(stats):
>   with(plots):
>   with(stats[statplots]):
>
> # (4) Draining time calculation
>      # Build a vector whose components are 293760 times reciprocal of flow input
>      # Set a functional operator to build
>      f := x -> 293760/x:
>     for counter from 1 to data_num do:
>      draining_time[counter]:= convert(f(flow[counter]),float):
>      # Draining_time has floating format
>     end do:
>
> # (5) Block to describe data
>     for counter from 1 to data_num do:
>       dt:=[seq(convert(f(flow[counter]),float),counter=1..data_num)]:
>       m:= describe[mean](dt):
>       Ku:=describe[kurtosis](dt):
>     end do:
>
>      print(`Mean of draining times: `, m):
>      print(`Kurtosis of draining times: `, Ku):
>
> # (6) Getting the minimun and maximun of draining times
>      for counter from 1 to data_num do:
```

```
>          if counter = 1 then:
>           draining_time_Min := draining_time[counter]:
>           draining_time_Max := draining_time[counter]:
>       else
>          if draining_time_Min > draining_time[counter] then:
>           draining_time_Min := draining_time[counter]:
>          end if:
>          if draining_time_Max < draining_time[counter] then:
>           draining_time_Max := draining_time[counter]:
>          end if:
>       end if:
>     end do:
>
>     print(`Minimum draining time: `, draining_time_Min):
>     print(`Maximun Draining time: `, draining_time_Max):
>
> # (7) Getting upper and lower limits of intervals
>       for k from 1 to intervals do:
>       # Lower and upper limits of k-interval
>         lower_limit[k] := draining_time_Min:
>         lower_limit[k] :=       lower_limit[k]+(k-1)*(draining_time_Max-draining_time_Min)/intervals:
>         upper_limit[k] := draining_time_Max:
>         upper_limit[k] := lower_limit[k]+(k)*(draining_time_Max - draining_time_Min)/intervals:
>         print(`Interval`, [`k`] , `(`,lower_limit[k],`,`,upper_limit[k],`)`):
>       end do:
>
> # (8) Draining time comparison and classification
>       for counter from 1 to data_num do:
>         finding := 0:
>         for k from 1 to intervals do:
>            if (draining_time[counter] > lower_limit[k]) and
>               (draining_time[counter] < upper_limit[k]) and
>               (finding = 0) then:
>        result[counter] := 204/(0.8*(lower_limit[k]+((upper_limit[k]-
lower_limit[k])/2))*sqrt(2*9.81*3)*60):
>          chain := cat(`Datum  `,convert(draining_time[counter],string)):
>          chain := cat(chain,` belongs to interval`, convert(k,string)):
>          chain := cat(chain,` [ `,convert(lower_limit[k],string),` , `,convert(upper_limit[k],string),` ]`):
>          chain := cat(chain,` and it is associated with `, convert(result[counter],string)):
>         print(chain):
>               finding := 1:
>            end if:
>         end do:
>         if finding = 0 then:
>            result[counter] := .11030318241:
>            chain := cat(`Datum `,convert(draining_time[counter],string),` does not belong to any interval`):
>           chain := cat(chain,` and is associated with `,convert(.1103031824,string)):
>          print(chain):
>         end if:
>     end do:
> # (9) Block to describe data
>      for counter from 1 to data_num do:
>         os:=[seq(result[counter],counter=1..data_num)]:
>         M:= describe[mean](os):
>         K:=describe[kurtosis](os):
>      end do:
```

```
>       print(`Outlet size range`, describe[range](os)):
>       print(`Mean of outlet sizes: `, M):
>       print(`Kurtosis of outlet sizes: `, K):
>
>
> # (10) Draining time vs outlet size plot
>      for counter from 1 to data_num do
>    points:={seq([convert(f(flow[counter]),float),result[counter]],counter=1..data_num)}:
>      end do:
>   pointplot(points,title=`draining time vs outlet size`);
>
> end proc;
```

Compilation

```
Outlet_size := proc(file, data_num, data_view, intervals::numeric, nonnegint)
local chain, imported_data, flow, counter, aux_counter, m, Ku, M, K, k, finding;
global f, dt, v, os, draining_time, draining_time_Min, draining_time_Max,
upper_limit, lower_limit, result, points;
    imported_data := Vector( 1 .. data_num );
    flow := Array( 1 .. data_num );
    draining_time := Array( 1 .. data_num );
    lower_limit := Array( 1 .. intervals );
    upper_limit := Array( 1 .. intervals );
    result := Array( 1 .. data_num );
    dt := Array( 1 .. data_num );
    v := Vector( 1 .. data_num );
    chain := `File from we take  `;
    chain := cat( chain, data_num, ` data:  ->  `);
    chain := cat( chain, file );
    print( chain );
    imported_data := ImportVector( file, format = rectangular );
    flow := imported_data;
    with( linalg );
    with( stats );
    with( plots );
    with( stats[ statplots ] );
    f := x → 293760×1/x;
    for counter to data_num do
        draining_time[ counter ] := convert( f( flow[ counter ] ), float )
```

```
end do ;
for counter to data_num do
        dt := [seq(convert( f(flow[counter]), float), counter = 1 .. data_num)];
        m := describe[mean](dt);
        Ku := describe[kurtosis](dt)
end do ;
print( `Mean of draining times: `, m);
print( `Kurtosis of draining times: `, Ku);
for counter to data_num do
        if counter = 1 then
                draining_time_Min := draining_time[counter];
                draining_time_Max := draining_time[counter]
        else
                if draining_time[counter] < draining_time_Min then
                        draining_time_Min := draining_time[counter]
                end if ;
                if draining_time_Max < draining_time[counter] then
                        draining_time_Max := draining_time[counter]
                end if
        end if
end do ;
print( `Minimum draining time: `, draining_time_Min);
print( `Maximun Draining time: `, draining_time_Max);
for k to intervals do
        lower_limit[k] := draining_time_Min;
        lower_limit[k] := lower_limit[k]
                + ((k − 1)×(draining_time_Max − draining_time_Min))/intervals;
        upper_limit[k] := draining_time_Max;
        upper_limit[k] := lower_limit[k]
                + k×(draining_time_Max − draining_time_Min)/intervals;
        print(Interval, [k], `(`, lower_limit[k], `, `, upper_limit[k], `)`)
end do ;
for counter to data_num do
        finding := 0;
        for k to intervals do
                if lower_limit[k] < draining_time[counter] and
                draining_time[counter] < upper_limit[k] and finding = 0 then
                        result[counter] := 4.250000000×
                                1/((1/2×lower_limit[k] + 1/2×upper_limit[k])×sqrt(58.86))
                                ;
```

```
                    chain := cat( `Datum  `, convert( draining_time[ counter ], string ) )
                        ;
                    chain := cat( chain, ` belongs to interval`, convert( k, string ) );
                    chain := cat( chain, ` [ `, convert( lower_limit[ k ], string ), ` , `,
                        convert( upper_limit[ k ], string ), ` ]` );
                    chain := cat( chain, ` and it is associated with `,
                        convert( result[ counter ], string ) );
                    print( chain );
                    finding := 1
                end if
            end do ;
            if finding = 0 then
                result[ counter ] := .11030318241;
                chain := cat( `Datum `, convert( draining_time[ counter ], string ),
                    ` does not belong to any interval` );
                chain := cat( chain, ` and is associated with `,
                    convert( .1103031824, string ) );
                print( chain )
            end if
        end do ;
        for counter to data_num do
            os := [ seq( result[ counter ], counter = 1 .. data_num ) ];
            M := describe[ mean ]( os );
            K := describe[ kurtosis ]( os )
        end do ;
        print( `Outlet size range`, describe[ range ]( os ) );
        print( `Mean of outlet sizes:  `, M );
        print( `Kurtosis of outlet sizes:  `, K );
        for counter to data_num do points := { seq(
            [ convert( f( flow[ counter ] ), float ), result[ counter ] ],
            counter = 1 .. data_num ) }
        end do ;
        pointplot( points, title = `draining time vs outlet size` )
    end proc
```

Program Execution

```
> Outlet_size("a:\\octnovdicdata.txt",92,1,4);
                     File from we take  92 data:  ->  a:\octnovdicdata.txt

Warning, these names have been redefined: anova, describe, fit, importdata, random, statevalf, statplots,
transform
```

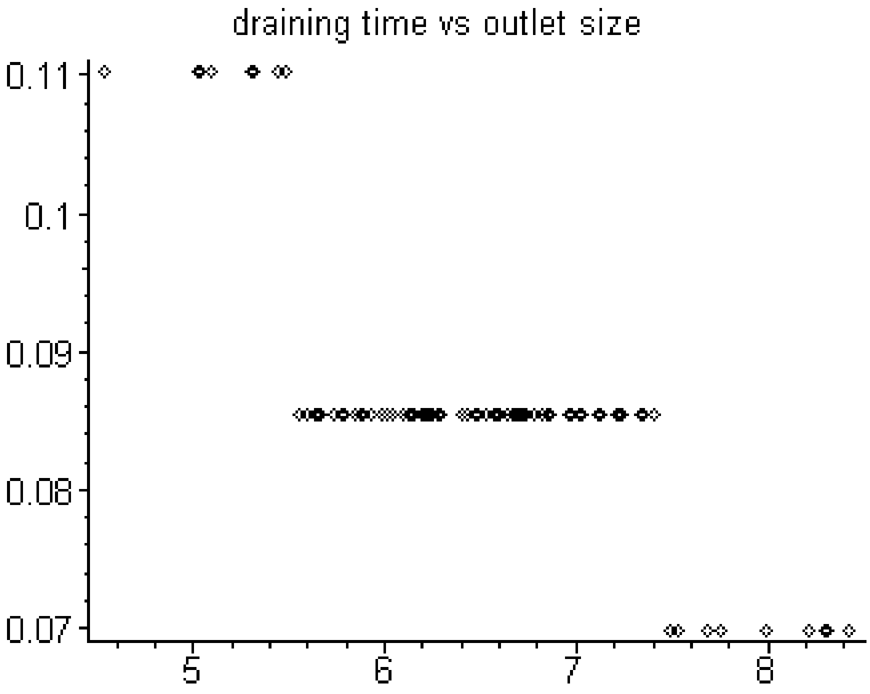
```
Warning, these names have been redefined: boxplot, histogram, scatterplot, xscale, xshift, xyexchange,
xzexchange, yscale, yshift, yzexchange, zscale, zshift
                    Mean of draining times: , 6.540653914
                  Kurtosis of draining times: , 3.012731889
                   Minimum draining time: , 4.535433071
                   Maximun Draining time: , 8.429268293
             Interval, [ 1 ], (, 4.535433071, ,, 5.508891876, )
             Interval, [ 2 ], (, 5.508891876, ,, 7.455809487, )
             Interval, [ 3 ], (, 6.482350682, ,, 9.402727100, )
             Interval, [ 4 ], (, 7.455809489, ,, 11.34964471, )
   Datum  5.930950939 belongs to interval2 [ 5.508891876 , 7.455809487 ] and it is ass\
       ociated with .8545673173e-1
   Datum 4.535433071 does not belong to any interval and is associated with .11030318\
       24
   Datum  5.645973477 belongs to interval2 [ 5.508891876 , 7.455809487 ] and it is ass\
       ociated with .8545673173e-1
```

et cetera[2]

```
   Datum  8.429268293 belongs to interval3 [ 6.482350682 , 9.402727100 ] and it is ass\
       ociated with .6974602342e-1
   Datum  7.353191489 belongs to interval2 [ 5.508891876 , 7.455809487 ] and it is ass\
       ociated with .8545673173e-1
   Datum  7.486238532 belongs to interval3 [ 6.482350682 , 9.402727100 ] and it is ass\
       ociated with .6974602342e-1
   Datum  6.731439047 belongs to interval2 [ 5.508891876 , 7.455809487 ] and it is ass\
       ociated with .8545673173e-1
            Outlet size range, .06974602342 .. .11030318241
                  Mean of outlet sizes: , .08573883853
                 Kurtosis of outlet sizes: , 5.715473113
```

---

[2] We omitted 85 result lines due to space reduction purpose.


draining time vs outlet size
0.11
0.1
0.09
0.08
0.07
5
6
7
8

>