Journal of Engineering and Computer Sciences
Qassim University, Vol. 4, No. 1, pp. 83-101 (January 2011/Moharram 1432H)

A Developed Softwae For an Improved Mesqa Hydraulic Design

Mohamed A. Nassar[*] and Abdelmoez. M. Hesham[**]
[*]*Assistant Professor, Water & Water Structures Engineering Dept., Faculty of Engineering, Zagazig University, Zagazig 44519, Egypt*
nasserzagazig@yahoo.com
[*] *Physics and Eng. Mathematics Dept., Mataria Faculty of Engineering, Helwan University, Cairo, Egypt, habdelmoez@yahoo.com*

*

(Received 29/7/2010; accepted for publication 8/1/2011)

Abstract. The proposed software is an advanced computational hydraulic software tool specially adapted for the design, and management of pressurized irrigation networks. The hydraulic solver uses specific strategies and incorporates several new features that improve the algorithms for pipe networks computation. These networks consist of a pumping station, main and secondary flow paths, valves, hydrants, and ancillary equipments. The software simulates hydraulically the flow in the pipe lines by computing the different nodes' pressure. It executes this throughout converting gis maps digitally into pipe lines, valves, and main pumping station. It is applied on the iip (i.e., irrigation improved project) which is one of the national irrigation development projects in egypt. There is no commercial program produces all these targets because iip has its own specific design criteria such as discharge, water duty, and different sets of pumping units. Total economic aspects including pump station and the network parts together with their constituents are computed too. Construction contracts according to the requirements of the egyptian irrigation ministry's specification are accomplished. It has been tested, calibrated and approved on some developed mesqas comparing manual and implemented results. The efficiency of some algorithms has been estimated using computational geometry rules.

Keyword: *hydraulics, gis, mesqa design criteria, vbasic, algorithm efficiency.*

1. Introduction

Reviewing the literature over the past five to ten years, there is indeed a substantial increase in the number of computer programs water-flow models. A number of packages are available that allow simulation models to be constructed for water criteria specific requirements. Popular packages include EPANET (US Environmental Protection Agency) [1], Infoworks (Wallingford software [2], and SynerGEE (Advantica) [3]. The design of distribution networks using these software packages has developed from trial and error to, more recently, the use of various forms of optimization, including genetic algorithms (e.g. Dandy et al., [4]). Bhattacharya et al. [5] proposed an ANN (Artificial Neural Network) with reinforcement learning which could learn to replicate the optimal control strategy (based on capturing operator experience). Rao and Salomons [6] developed a GA (Genetic Algorithm) and an ANN model for capturing the knowledge base of an EPANET model and consequently producing a near optimal solution.

This paper presents a software to simulate the pressurized pipe system. The software may be as simple pipe carrying water from one reservoir through network to one valve, or it may be very complex with many interconnected pipes that distribute the flow throughout a large pipe networks. Our software can model pipe networks and calculate the flow and pressure throughout a system with different pipe sizes and pipe materials, supply and discharge tank, pumps, valves, flow controls, system demands and other component. The pipeline system is modeled by drawing the join points and the connecting pipes on a drawing pane. Horizontal, vertical or sloping lines can be used to connect one node to another node. The optimum design is fulfilled throughout dividing the design process into its main categories and precisely defining each category. Section 3 provides criteria for design of water distribution systems, while section 4 gives the proposed program calibration utilities including a handout solved example on these criteria. Section 5 gives more sophisticated pictures of the proposed software results, while section 6 concludes and suggests the future work of this paper.

2. Software Overview

The program will allow user to draw a complex pipeline system and analyze the features of the system when flow is occurring. It calculates the balanced steady flow and pressure conditions of the system. In addition, it will allow user to perform analysis of alternate systems under various operating conditions. The physical data describing the system is entered by the user and typically includes:

- The internal size, internal roughness and length of each pipe.
- The elevation of each pipe join point (node) and the In-flow and the Out-flow at each join point.
- The elevation, liquid level and surface pressure data for the main tank.

The reported results include: flow rates for each pipe; pressures at each node; HGL (hydraulic grade line) values; pump operating points and NPSH (i.e., Net Positive Suction Head) at pump inlet. Total economic aspects including pump station and the network parts together with their constituents are computed too.

Construction contracts according to the requirements of the Egyptian irrigation ministry's specification are accomplished.

In the other hand, under IIP (i.e., Irrigation Improved Project), hydraulic design of improved branch canals has been carried out using a version of Mott MacDonald’s in-house simulation model, which was specially customized at the start of the project in 1996/97 to meet the specific requirements of IIP. This version of the model runs under DOS, which is not now available on most computers. In 1993, Delft Hydraulics introduced a new unsteady-flow simulation software package, SOBEK [7]. It includes a link to MATLAB so that control decisions can be made within that framework. Water levels are passed to MATLAB and gate position changes are passed back to SOBEK. The control routines are written as MATLAB files. Recently, canal control studies have been conducted with the SOBEK–MATLAB combination by Delft Hydraulics.

Actually, the design staff of IIP (i.e., Irrigation Improved Project) was suffering from a big problem, which is the difficulty to abstract the maximum water levels at the different studied point in the same reach. We present here a simple program to abstract these values in easy way. The main purpose is simplifying the way in which the design staff defines the maximum water levels. We added a subroutine to the present software, which helps the user to abstract the water levels and save them as an excel file.

3. Criteria of the Design of Water Distribution Systems

For the IIP-area, the design criteria, which are respected during the development of the software are as the followings:

- Rice (maximum crop water requirement) could be grown in 100% of the mesqa command area.
- Maximum number of pumping units to be 3-units per pump house.
- Mesqas will be designed using PVC pipes 4 bar pressure rating.
- The minimum pipe diameter used for mesqa pipelines will be 200 mm.
- In case of using a stand, the pipeline will be provided with an open air vent at the end of each branch.
- In case of using direct connection, the pipeline will be provided with an air/vacuum valve on the pump delivery manifold and a pressure relief valve at the end of the pipeline.
- The maximum water velocity in the pipeline should not exceed 1.50 m/sec,
- The head loss through the mesqa pipeline network is to be determined according to the following empirical formula:

$$H_L = \left(c_{in} + n_T K + n_{BB} c\right)\frac{V_P^2}{2g} + 3.29\frac{V_R^2}{2g} + H_f + H_{marwa} + \text{Min Valve Head} \quad (1)$$

where: c_{in} is the head loss coefficient at mesqa pipeline inlet, $c_{in} = 0.50$, K is the head loss coefficient at Tee connection, see table No. (1), n_T is the number of the Tee along mesqa pipeline, c_B is the head loss coefficient in bends, $c_B = 0.90$; n_B is the number of the bends along mesqa pipeline, V_p is the average water velocity in the mesqa pipeline; V_R is the average velocity in the riser (m/sec); H_f is the friction head loss, it is computed using William-Hazen's equation,

$$H_f = (3.59/C_H)^{1.852} (Q_m^{1.852} / D_P^{4.87}) L \quad (2)$$

where: Q_m is the discharge (m^3/sec), D_P is the pipe diameter (m), L is the pipeline length (m), and C_H is Hazen friction Coef., 150 and 140 for PVC and PE, respectively, H_{marwa} is the operating head at the marwa off take ranges from 1.5 to 2.0 m [used for case of neglecting the marwa during design stage].

Table (1). Head loss coefficient due to the Tee connection with valves

Diameter of the pipeline (mm)	*K*
200 to 250	0.28
300 to 400	0.26
450 to 600	*0.24*

4. Calibration of the Proposed Software

4.1. Practical calibration:

The developed software is calibrated using a simple type of networks. It simulates the pressurized flow in a simple network. It includes a single path only, see Fig. (1). It is necessary to mention that it is an imaginary network.

4.1.1. Design example:

4.1.1.1. Given:

A parcel of land of about 43F has a main canal, a pump station, and 5 valves. All needed data for design are presented as tabulated in table (2).

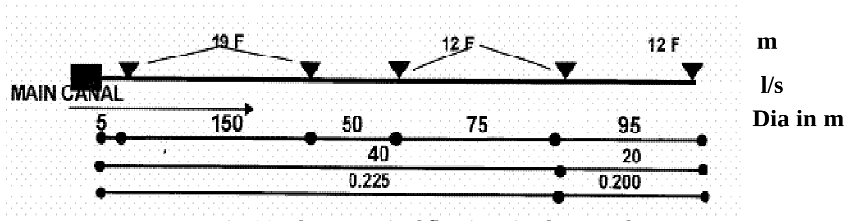


Fig. (1). The pressurized flow in a simple network.

Table (2). Designed data for the simple network

Km	**0**	**0.005**	**0.155**	**0.205**	**0.28**	**0.375**
Land level	**2.54**	**2.41**	**2.41**	**2.43**	**2.37**	**2.61**
land level for marwa		**2.33**	**2.33**	**2.31**	**2.35**	**2.42**

4.1.1.2. Required:

The main targets of the design are including both operating head (m), and Stand total height (m).

4.1.1.3. Manual solution:

- Max. Discharge = 0.84*43 = 36.12 1/s ≅ 40 l/s.
- Max no of valves working in certain time = 2 valves.
- $H_L = (c_{in} + nK_T + nc_{BB})\frac{V_P^2}{2g} + 3.29\frac{V_R^2}{2g} + H_f + H_{marwa}$ +0.6+(if tank >6.0 take 0.2 into account)
- $H_{L(0)}$ = (0.5 + n x0.28 +0.9 x1) $Vp^2/2g$ +3.29(V_R^2)/2g +(3.59/150) $^{1.852}$ x ($Q^{1.852}/D^{4.87}$) x L + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0)}$ = (0.5+ 2 x 0.28 + 0.9 x 1) $(1.09)^2/2g$ +3.29(1.0^2)/2g +(3.59/ 150) $^{1.852}$ x($0.04^{1.852}$/($0.216)^{4.87}$)x 280 +(3.59/150) $^{1.852}$ x ($(0.02)^{1.852}/(0.192)^{4.87}$)x 95+1.5 + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0)}$ = 0.1186 +0.1676 +1.2511+0.2086+1.5+0.6= 3.84616m
- $H_{L(0.005)}$ = $\underline{(0.5+ 2 \text{ x } 0.28 + 0.9 \text{ x } 1) (1.09)^2/2g}$ +$\underline{3.29(1.0^2)/2g}$ $\underline{+(3.59/ 150)^{1.852} \text{ x}(0.04^{1.852}/(0.216)^{4.87})\text{x } 275}$ + $\underline{(3.59/150)^{1.852} \text{ x } ((0.02)^{1.852}/(0.192)^{4.87})\text{x } 95}$+1.5 + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0.005)}$ = 0.1186 +0.1676 +1.228+0.2086+1.5+0.6= 3.826m
- $H_{L(0.155)}$ = $\underline{(0.5+ 2 \text{ x } 0.28 + 0.9 \text{ x } 1) (1.09)^2/2g}$ +$\underline{3.29(1.0^2)/2g}$ $\underline{+(3.59/ 150)^{1.852} \text{ x}(0.04^{1.852}/(0.216)^{4.87})\text{x } 125}$ + $\underline{(3.59/150)^{1.852} \text{ x } ((0.02)^{1.852}/(0.192)^{4.87})\text{x } 95}$+1.5 + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0.155)}$ = 0.1186 +0.1676 +0.558+0.2086+1.5+0.6= 3.15m
- $H_{L(0.205)}$ = $\underline{(0.5+ 2 \text{ x } 0.28 + 0.9 \text{ x } 1) (1.09)^2/2g}$ +$\underline{3.29(1.0^2)/2g}$ $\underline{+(3.59/ 150)^{1.852} \text{ x}(0.04^{1.852}/(0.216)^{4.87})\text{x } 75}$+ $\underline{(3.59/150)^{1.852} \text{ x} ((0.02)^{1.852}/(0.192)^{4.87})\text{x } 95}$+1.5 + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0.205)}$ = 0.1186 +0.1676 +0.335+0.2086+1.5+0.6= 2.92
- $H_{L(0.280)}$= $\underline{(0.5+ 2 \text{ x } 0.28 + 0.9 \text{ x } 1)}$ $\underline{(1.09)^2/2g}$ $\underline{+3.29(1.0^2)/2g}$$\underline{+(3.59/150)^{1.852} \text{ x } ((0.02)^{1.852}/(0.192)^{4.87})\text{x } 95}$+1.5 + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0.280)}$ = 0.1186 +0.1676 +0.2086+1.5+0.6= 2.59
- $H_{L(0.375)}$= $\underline{(0.5+ 2 \text{ x } 0.28 + 0.9 \text{ x } 1) (1.09)^2/2g}$ +$\underline{3.29(1.0^2)/2g}$+1.5 + 0.6 +(if tank >6.0 take 0.2 into account)
- $H_{L(0.375)}$ = 0.1186 +0.1676 +1.5+0.6= 2.38

Finally, the hydraulic calculation can be tabulated in the following table (3) and figure (4).

- Stand total height = 3.8461 + 0.75+0.4+.25= <u>5.2461</u> m
- Stand Top level = 5.2461 + 1.09= <u>6.336</u> m.

Table (3). Manual design operating head for the different nodes in the simple network

Km	0	0.005	0.155	0.205	0.280	0.375
Land level	**2.54**	**2.41**	**2.41**	**2.43**	**2.37**	**2.61**
Op. head (m)	**3.84**	**3.82**	**3.15**	**2.92**	**2.59**	**2.38**

4.1.2. Solution of the simple network using the proposed software:

The simple network is modeled using the proposed software in three steps as following:

1- Input the data of the design including area served by the pump station; the land level the pump station; and the necessary definition of the network.

2- Digitizing the network path as shown in Fig. (2A), and checking the input date for the path itself, see Fig. (2B), and.

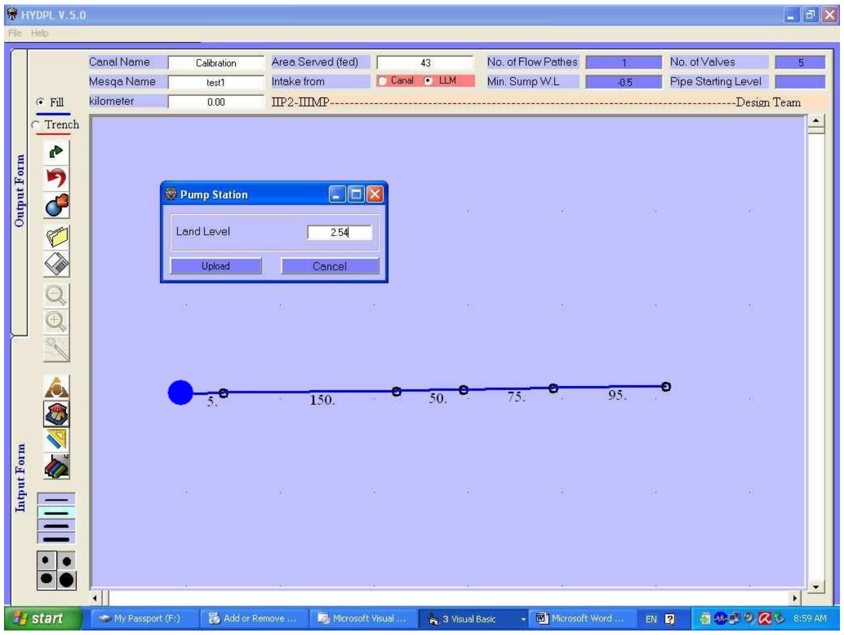
3- Presenting the output data including the design for the path and pump sets and both operating head (m), and Stand total height (m), see Fig. (3).

It can be noticed that, the out screen of the software including the all design data for the flow paths and network see, Fig. (3) at the left hand side. In addition, the hydraulic gradient line and all graphs for the designed path are presented in the same figure. The output of the software can be listed as following:

- Stand height (m)= 5.270;
- Q (l/s)=40 ;
- Mesqa Length (m)= 364.3;
- Length per fed.(m)= 15.839;
- Pump Cat. = Categ. 2 cover (5.0 to 8.5 m); and
- Pump sets (i.e., the formulation of the pumps): two pumps of 20(lit/s)

4.1.3. Analysis of practical calibration:

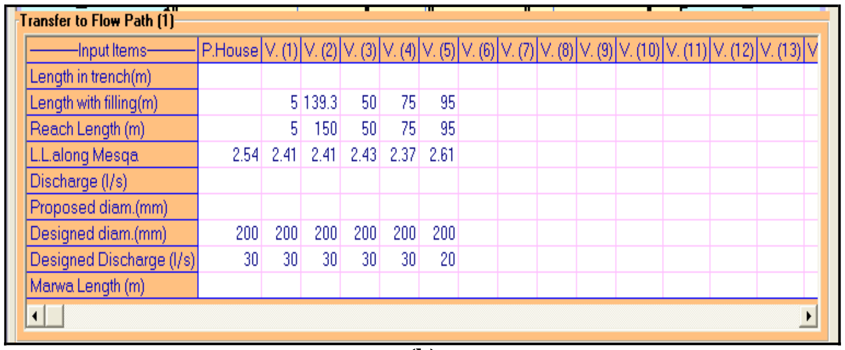
Figure (4) shows a comparison between the hydraulic gradient line calculated with the developed software and that calculated manually for the studied network. It can be seen that the hydraulic gradient line calculated from the software is higher than that calculated manually by about 18cm, see Fig.(4). The calculated stand height using the proposed software is nearly identical with that calculated by the manual procedure. The difference was about 13.1 cm. The statistics' measures, which are R2 and correlation factor, are used to measure the ability of the developed software as a tool to calculate the hydraulic gradient line through the studied network. R^2 and correlation factor are 0.978 and 0.989, respectively. Using statistical analysis principles, and based on both modeled and calculated values of the hydraulic gradient line, it was found that the developed software is well verified and an accurate tool to the hydraulic gradient line along the studied network. The accuracy was more than 97%.



Canal Name: Calibration
Area Served (fed): 43
No. of Flow Pathes: 1
No. of Valves: 5
Mesqa Name: test1
Intake from: Canal [✓ LLM]
Min. Sump W.L: -0.5
Pipe Starting Level:
kilometer: 0.00
[✓ Fill] Trench
IIP2-IIIMP---Design Team

Pump Station
Land Level: 2.54
Upload Cancel

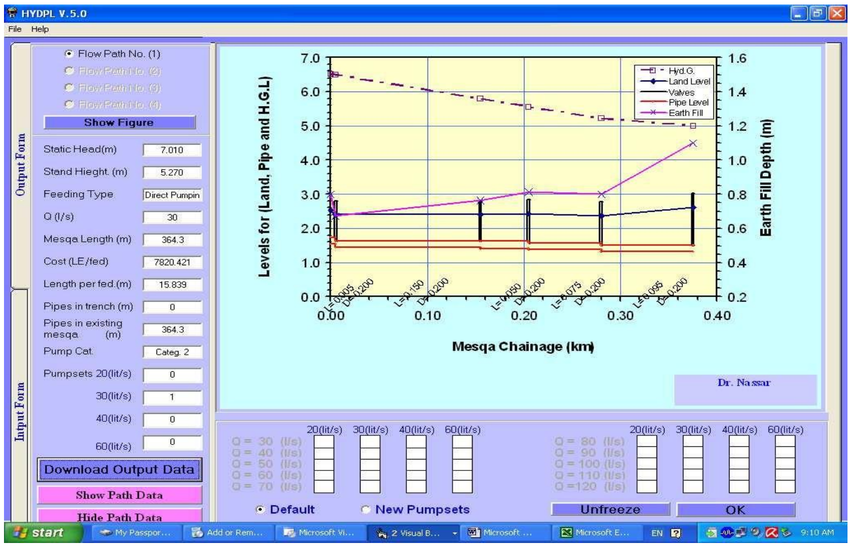
(a)


Transfer to Flow Path (1)

Input Items	P.House	V. (1)	V. (2)	V. (3)	V. (4)	V. (5)	V. (6)	V. (7)	V. (8)	V. (9)	V. (10)	V. (11)	V. (12)	V. (13)	V
Length in trench(m)															
Length with filling(m)		5	139.3	50	75	95									
Reach Length (m)		5	150	50	75	95									
L.L.along Mesqa	2.54	2.41	2.41	2.43	2.37	2.61									
Discharge (l/s)															
Proposed diam.(mm)															
Designed diam.(mm)	200	200	200	200	200	200									
Designed Discharge (l/s)	30	30	30	30	30	20									
Marwa Length (m)															

(b)

Fig. (2). The inputs screen shots of the software; (a) the digitizing form (b) digitized data.



[✓ Flow Path No. (1)] Flow Path No. (2) Flow Path No. (3) Flow Path No. (4)

Static Head(m)	7.010
Stand Hieght. (m)	5.270
Feeding Type	Direct Pumpin
Q (l/s)	30
Mesqa Length (m)	364.3
Cost (LE/fed)	7820.421
Length per fed.(m)	15.839
Pipes in trench (m)	0
Pipes in existing mesqa (m)	364.3
Pump Cat.	Categ. 2
Pumpsets 20(lit/s)	0
30(lit/s)	1
40(lit/s)	0
60(lit/s)	0

Fig. (3). The outputs screen shot of the software for the simple network.

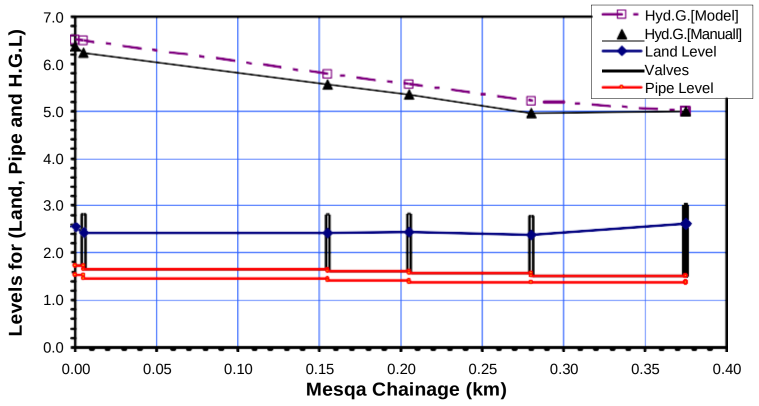


Fig. (4). Calculated and modelled hydraulic gradient line through the studied network.

4.2. Theoretical Calibration

4.2.1. The Efficiency of some algorithms included:

The algorithm efficiency can be measured either by the time it takes to run a program or by the space the program takes up in memory [8]. We will focus on an algorithm's efficiency with respect to time. How do we compare the time efficiency of two algorithms that solve the same problem? One approach: implement the two algorithms in Visual Basic (VB) and run the programs … this presents some difficulties:

i) How are the algorithms coded?

If algorithm A runs faster than algorithm B, it will obviously give one algorithm an advantage. Hence, we would be comparing the algorithms' implementations, rather than the algorithms themselves. We should not compare implementations, because they are sensitive to factors, such as programming style, that tend to cloud the issue of which algorithm is inherently more efficient.

ii) What computer speed should we use?

If one computer is faster than the other, it will obviously give one algorithm an advantage. Hence, both algorithms should be run on the same computer.

iii) What computer type should we use?

The type of computer is also more important. The particular operations that the algorithms require can cause A to run faster than B on one computer while the opposite is true on another computer. Hence, we must be able to compare efficiencies independent of a particular computer.

iv) What data should the program use?

There is always the danger of selecting data sets for which one of the algorithms runs uncharacteristically fast (or slow). Example: sequential search vs. binary search when search item is the first element in the array. Hence, we must be able to compare efficiencies independent of a particular data set. To overcome these difficulties, computer scientists employ mathematical techniques that analyze algorithms independent of specific implementations, computers, and data [9, 10]. An algorithm's efficiency is related to the number of operations it requires. If a function contains no loops, its efficiency is simply a function of the number of instructions it contains. In other words, the more instructions the function contains, the longer it will take to execute. That being said, with current computer speeds on the order of 3 GHz, it doesn't make much of a difference if our program has 10 instructions or 1000 instructions … we really won't notice a difference. However, when we are dealing with functions that loop, the problem becomes non-trivial. The study of algorithm efficiency therefore focuses on loops.

We typically discuss an algorithm's efficiency as a function of the number of elements to be processed. For example, if we want to determine the efficiency of an array-sorting algorithm, we express the efficiency of the algorithm as a function of the number of elements of the array, *n*. The general format is: f (n) = efficiency.

Let us start with a simple loop. We want to know how many times the body of the loop is executed

In the following code:

```
i=1
  loop (i<=1000)
           (loop body)
            i = i + 1
  end loop
```

The answer is 1000 times. The number of iterations is directly proportional to the loop factor, 1000.

Hence, the larger the loop factor, the more loop iterations we will complete. Because the efficiency is directly proportional to the number of iterations, its efficiency can be represented as: f(n) = n. However, the answer is not always as straightforward as it was in the previous example. For example, consider the following loop:

```
i = 1
loop (i <= 1000)
        (loop body)
        i = i + 2
end loop
```

In this case, the body of the loop is executed 500 times – half the value of the loop factor. Once again, however, the larger the loop factor, the more loops we execute. The efficiency of this loop is therefore:

f(n) = n/2. If you were to plot either of these loop efficiencies, you would get a straight line. Hence, they all called *linear loops*.

In our previous examples, our control variable was incremented by either 1 or 2 each time through the loop. Now consider a loop in which the controlling variable is multiplied or divided in each loop.

```
        Multiply Loops
        i = 1
        loop (i < 1000)
                (loop body)
                i = i * 2
end loop
        Divide Loops
        i = 1000
                loop (i >= 1)       (loop body)
                i = i / 2
end loop
```

How many times are the loop bodies executed in the above code sections? See table (4).

As we can see, the number of loop iterations is 10 in both cases. Note that in each iteration the value of i doubles for the multiply loop and is cut in half for the divide loop. Thus, the number of iterations of the loop is a function of the multiplier or divisor, in this case, 2. That is, the loop continues while the Condition shown below is true:

multiply: $2^{\text{iterations}} < 1000$, divide: $1000 / 2^{\text{iterations}} >= 1$

Generalizing the analysis, we can say that the efficiency of loops that multiply or divide by 2 is determined by the following formula: $f(n) = \log_2 n$.

For linear logarithmic loops, the following code will be considered:

```
        i = 1
loop (i <= 10)
```

```
        j = 1
        loop (j <= 10 )
                (loop body)
                j = j * 2
        end loop
        i = i + 1
end loop
```

Table (4). Logarithmic loops.

Multiply Loop		Divide Loop	
Iteration	Value of i	Iteration	Value of i
1	1	1	1000
2	2	2	500
3	4	3	250
4	8	4	125
5	16	5	62
6	32	6	31
7	64	7	15
8	128	8	7
9	256	9	3
10	512	10	1
(exit)	1024	(exit)	0

The inner loop is a multiply loop. The number of iterations in the inner loop is therefore $\log_2 10$. We must then multiply this by the number of times the outer loop executes. This gives us $10*\log_2 10$ which is generalized as $f(n) = n\log_2 n$.

With the speed of computers today, we are not concerned with an exact measurement of an algorithm's efficiency as much as we are with its general order of magnitude [11, 12]. For example, if the analysis of 2 algorithms shows that one finishes after 15 iterations while the other takes 25 iterations; then they are both so fast that we can't see the difference. However, if one algorithm finishes after 15 iterations and the other takes 15,000 iterations, this is a more significant difference.

We have shown that the number of iterations an algorithm executes, f(n), can be expressed as a function of the number of elements associated with the algorithm. Although the efficiency equation derived for a function can be complex, we can examine the dominant factor in the equation to determine the relative magnitude of the efficiency.

Hence, we don't need to determine the complete measure of efficiency, only the factor that determines the magnitude. This factor is the *big-O*, as in "on the order of," and is expressed as O (n). This simplification of efficiency is known as *big-O notation*. The big-O notation can be derived from
f (n) using the following steps:

1- We set the coefficient of each term to 1.

2- We keep the largest (least efficient) term in the function and discard the others.

Terms are ranked from most efficient (leftmost terms) to least efficient (rightmost terms) as shown below:

constants $\log_2 n$ n $n\log_2 n$ n^2 $n^3 \ldots n^k$ 2^n n!
For example, let's calculate the big-O notation of the following efficiency:

$$f(n) = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$$

First, we set all coefficients to 1. This gives us: $\boldsymbol{n^2 + n}$

Next, we keep the largest term in the function and discard all others. This leaves: n^2. Therefore, we can write the big-O notation as: $O(n^2)$.

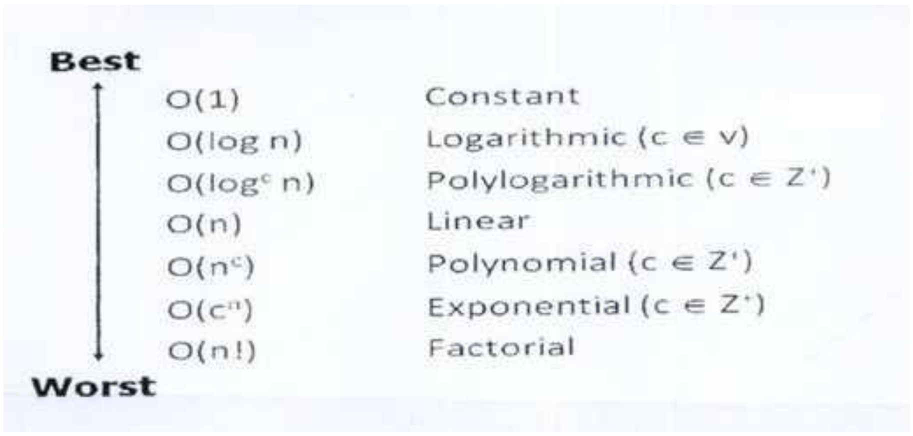
As another example, let's look at the polynomial expression:

$$f(n) = 6n^4 \log n + 12n^3 + 2n^2 + n + 128$$

First, we eliminate all of the coefficients: $f(n) = n^4 \log n + n^3 + n^2 + n + 1$

We then select the largest term and discard the rest. This gives us: $O(n^4 \log n)$

Note that constants are the MOST efficient. We think about it like this: If we have an efficiency of 100,000,000 vs. n, which is more efficient? Answer: since n can equal 100,000,001, the constant is the most efficient. See the following criteria:



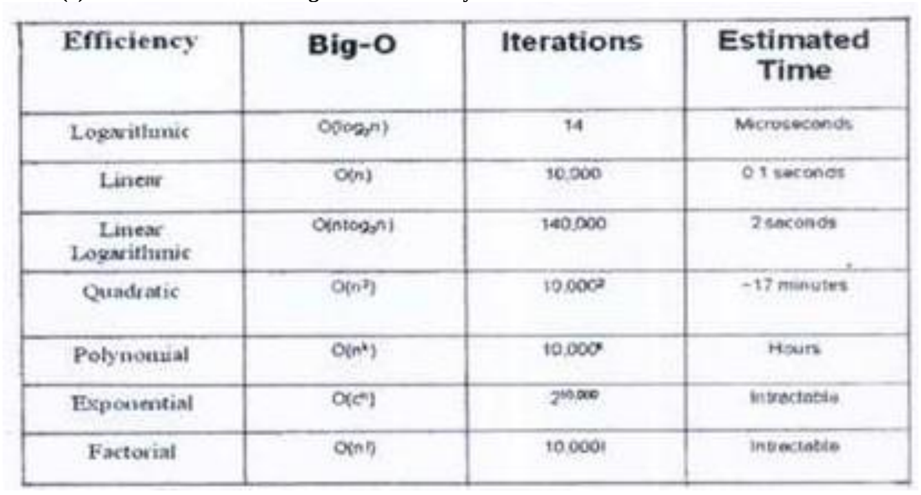
Best		
↑	$O(1)$	Constant
	$O(\log n)$	Logarithmic ($c \in V$)
	$O(\log^c n)$	Polylogarithmic ($c \in Z^+$)
	$O(n)$	Linear
	$O(n^c)$	Polynomial ($c \in Z^+$)
	$O(c^n)$	Exponential ($c \in Z^+$)
↓	$O(n!)$	Factorial
Worst		

When comparing two algorithmic efficiencies, the "most efficient" one grows the slowest. Easy test: After computing the efficiencies (e.g., $n^{0.5}$ and n^2) plug in a huge value for n; then whichever efficiency results in the smallest value is the most efficient. Here $n^{0.5}$ is more efficient than n^2.

To get a feel for how much of a difference an algorithm's efficiency makes, check out the table (5) together with figure (5). The table assumes an instruction speed of 1 microsecond, 10 instructions per loop, and n=10,000.

Note: performing an order-of-magnitude analysis implicitly assumes that the algorithm will be used to solve large problems.

Table (5). Standard measure of algorithm efficiency.



Efficiency	Big-O	Iterations	Estimated Time
Logarithmic	$O(\log_2 n)$	14	Microseconds
Linear	$O(n)$	10,000	0.1 seconds
Linear Logarithmic	$O(n\log_2 n)$	140,000	2 seconds
Quadratic	$O(n^2)$	$10{,}000^2$	~17 minutes
Polynomial	$O(n^k)$	$10{,}000^k$	Hours
Exponential	$O(c^n)$	$2^{10{,}000}$	Intractable
Factorial	$O(n!)$	10,000!	Intractable

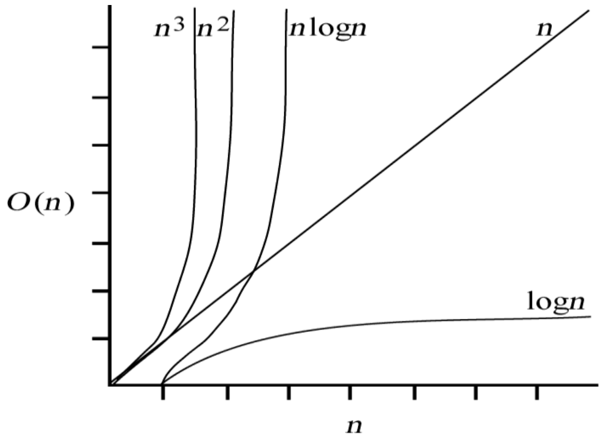


Fig. (5). Standard measure of algorithm efficiency.

4.2.2. Analysis of theoretical calibration:

Algorithm analysis is the area of computer science that provides tools for comparing the efficiency of different methods of solution. This analysis concerns itself primarily with *significant* differences in efficiency. Usually, significant differences only arise through superior solutions and rarely through clever coding tricks. Reductions in computing costs due to clever coding tricks are often more than offset by reduced program readability, which increases human costs.

An algorithm analysis should focus on gross differences in the efficiency of algorithms that are likely to dominate the overall cost of a solution. Otherwise, we could select an algorithm that runs a fraction of a second faster than another algorithm yet requires many more hours of our time to implement and maintain.

In our proposed software there are many algorithms. One of the very beginning ones are that for transporting design data for every path of the network paths under study from the program design file (since the program carries out the design operations in a hidden Excel file) and showing them to the designer on the screen as a table form to ensure the input data. This algorithm contains the following iterations:

```
For i = 3 To 25
iii = i - 2
If Option1.Value = True Then grdRunData.TextMatrix(7, iii) = C.Cells(18, i).Value
If Option2.Value = True Then grdRunData.TextMatrix(7, iii) = C.Cells(28, i).Value
If Option3.Value = True Then grdRunData.TextMatrix(7, iii) = C.Cells(38, i).Value
If Option4.Value = True Then grdRunData.TextMatrix(7, iii) = C.Cells(48, i).Value

If Option1.Value = True Then grdRunData.TextMatrix(8, iii) = C.Cells(60, i).Value
If Option2.Value = True Then grdRunData.TextMatrix(8, iii) = C.Cells(61, i).Value
If Option3.Value = True Then grdRunData.TextMatrix(8, iii) = C.Cells(62, i).Value
If Option4.Value = True Then grdRunData.TextMatrix(8, iii) = C.Cells(63, i).Value

Next
```

The iterations start from 3 to 25, i.e. 23 points, which is the maximum number of points on the same path. Every iteration contains 4 paths as a maximum allowed number of paths. Each path has two data entries that are final diameters in mm and the discharge in l/s. As we see in that loop, the efficiency of this loop is *O (n)* which is a constant efficiency. It is the best algorithm efficiency.

Another loop is that creates virtual pumps formulations. This means that the total pump station discharge has to be partitioned into smaller pumps formulation components that are available in the execution contracts. As example, if the total pump station discharge is 100 l/s and the contract allows only 40 and 60 l/s pumps, then it is obvious to use these two discharge pumps formulation components. These virtual formulations are come from both experience and user demands.

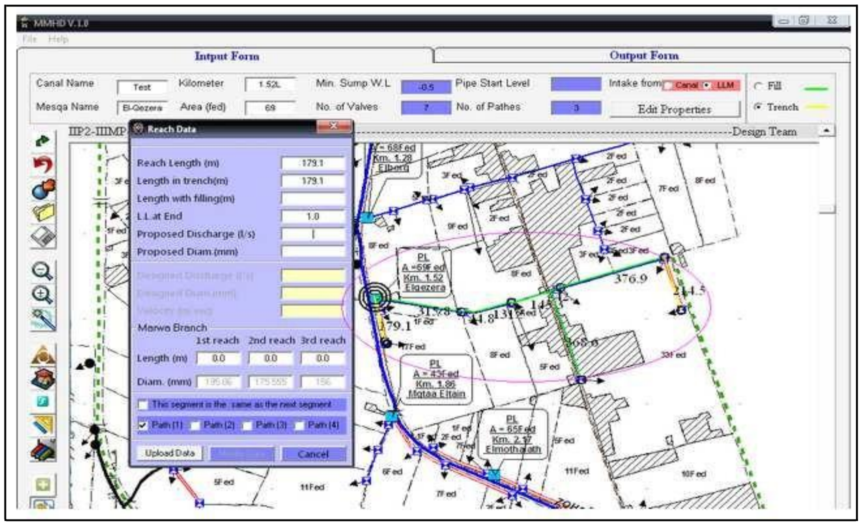
```
For gf = 0 To 3
If C.Cells(13, 8).Value <= 30 Then Text40(gf).Text =C.Cells(13, 9+ gf).Value Else
Text40(gf).Text"=
If C.Cells(13, 8).Value <= 40 And C.Cells(13, 8).Value > 30 Then Text40(gf).Text =
C.Cells(13, 9 + gf).Value Else Text40(gf + 4).Text"=
If C.Cells(13, 8).Value <= 50 And C.Cells(13, 8).Value > 40 Then Text40(gf +
8).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 8).Text"=
If C.Cells(13, 8).Value <= 60 And C.Cells(13, 8).Value > 50 Then Text40(gf +
12).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 12).Text"=
If C.Cells(13, 8).Value <= 70 And C.Cells(13, 8).Value > 60 Then Text40(gf +
16).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 16).Text"=
If C.Cells(13, 8).Value <= 80 And C.Cells(13, 8).Value > 70 Then Text40(gf +
20).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 20).Text"=
If C.Cells(13, 8).Value <= 90 And C.Cells(13, 8).Value > 80 Then Text40(gf +
24).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 24).Text"=
If C.Cells(13, 8).Value <= 100 And C.Cells(13, 8).Value > 90 Then Text40(gf +
28).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 28).Text"=
If C.Cells(13, 8).Value <= 110 And C.Cells(13, 8).Value > 100 Then Text40(gf +
32).Text = C.Cells(13, 9 + gf).Value Else Text40(gf + 32).Text"=
If C.Cells(13, 8).Value > 110 Then Text40(gf + 36).Text = C.Cells(13, 9 + gf).Value
Else Text40(gf + 36).Text"=

Next
```

As we see in that loop, the efficiency of this loop is *O (n)* which is a constant efficiency. It is the best algorithm efficiency. The following is a group of loops that are designed efficiently to fulfil the requirements of the excel file. The later contains the design properties such as the internal diameter for every pipe type, the design water duty, pipe class, … etc. It transports them from the design properties file to the designer on the screen in order to accept, modify, or cancel if it is required.

```
iii = 0

For iii = 1 To 9
Text13 (iii - 1).Text = C.Cells(iii + 9, 7).Value
Next

iii = 0
For iii = 9 To 17
Text13 (iii).Text = C.Cells(iii + 1, 11).Value
Next

iii = 0
For iii = 18 To 25
Text13 (iii).Text = C.Cells(iii - 15, 20).Value
Next

iii = 0
For iii = 26 To 29
Text13 (iii).Text = C.Cells(iii + 11, 5).Value
Next
```

After the theoretical calibration has been achieved one can conclude that, all the algorithms of the presented software are efficient more than 50% and have the best quality. This is because the program fulfils time requirement, which is half the way to the complete algorithm efficiency. The rest of the efficiency of the algorithms which is the memory requirement is about 40 % achieved. This is because the reserved memory size for the proposed software constants and variables are so minimized that the available memory size for the user is huge. As a result, the algorithm efficiency is more than 90 %.

5. Snap Shots of Some Sophisticated Field Solved Problems


MMHD V.1.0

File Help

Intput Form | Output Form

Canal Name: Test | Kilometer: 1.52L | Min. Sump W.L: -0.5 | Pipe Start Level: | Intake from: Canal [✓ LLM] | Fill
Mesqa Name: El-Gezera | Area (fed): 69 | No. of Valves: 7 | No. of Pathes: 3 | Edit Properties | [✓ Trench]

Reach Data

Reach Length (m): 179.1
Length in trench(m): 179.1
Length with filling(m):
L.L.at End: 1.0
Proposed Discharge (l/s):
Proposed Diam.(mm):

Marwa Branch

	1st reach	2nd reach	3rd reach
Length (m)	0.0	0.0	0.0
Diam. (mm)	195.06	175.555	156

This segment is the same as the next segment

[✓ Path (1)] Path (2) Path (3) Path (4)

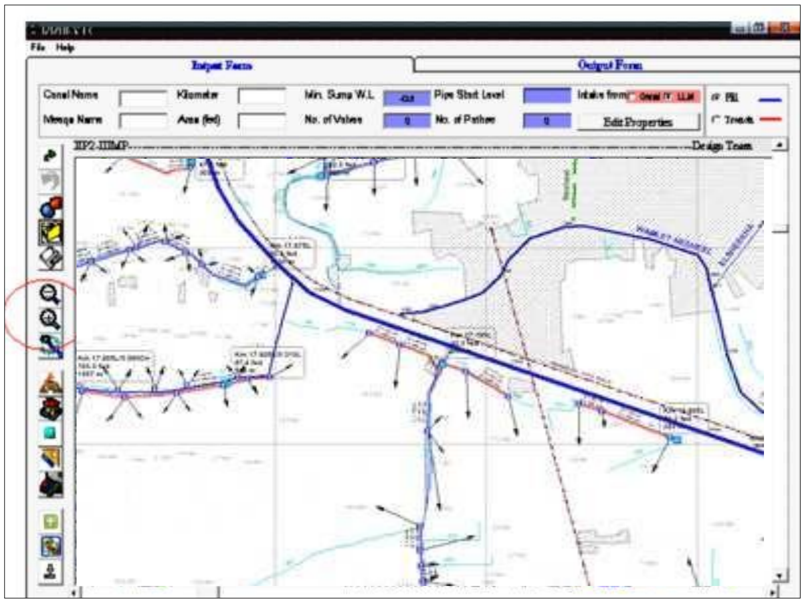


Figure (6). Two selected screen shots of the software for the complicated networks

6. Conclusions

We have proposed software for fully integrated design framework of a multi-disciplinary approach and cost computations for pressurized pipe lines used in IIP projects. It can carry out many tasks such as:

- Simulations of flows through the pipe lines implemented in IIP projects,
- Estimates the cost for the all elements of the pipe lines including the pump stations,
- Prepare the engineering estimates and Arabic version of the contract which is necessary to biding stage,
- It includes new features to assist Sobek software to extract the date about the longitudinal sections,
- It gives the ability to users to detect the maximum and minimum water levels and
- It prepares the necessary files to change all the abstraction points in any Sobek case.

It has been designed for experts and non-experts alike. The framework is composed of several modules, grouped around a mutual interface, while being capable to interact with one another. Some of the used algorithms in the presented software are reviewed against efficiency. The proposed program is calibrated both practically and theoretically.

In the future, it may add a module to the software to prepare the necessary Auto Cad files for layout and longitudinal section of the available paths of the designed mesqa,

7. References

[1] Lewis A. Rossman, "EPANET 2-USERS MANUAL", *National Risk Management Research Laboratory, Office of research and development, U.S. Environmental Protection Agency, Cincinnati, Oh 45268*, (2000).

[2] Wallingford Software, "*InfoWorks CS Technical Review*", (2009).

[3] StonerSofware, "*SynerGEE – ESRI, For City of Leesburg Integration*", (2008).

[4] Dandy, G. A., Simpson, A. R., and Murphy, L. J. "An improved genetic algorithm for pipe network optimization", *Water Res.,* Vol. 32, No. 2, (1996), pp. 449–458.

[5] Bhattacharya, B., Lobbrecht, A. H., and Solomatine, D. P. "Neural networks and reinforcement learning in control of water systems", *J. Water Res. Pl.-ASCE*, Vol. 129, No. 6, (2003), 458–465.

[6] Rao, Z. and Salomons, E. "Development of a real-time, nearoptimal control process for water distribution networks", *J. Hydroinform., IWA Publishing*, Vol. 9, No. 1, 2007), pp. 25–37.

[7] Delft Hydraulics LTD, The Independent Consulting and Research Institute, "*SOBEK210, SOBEK Software and Manuals*", Netherlands, http://www.wldelft.nl and http://www.SOBEK.nl, (2006).

[8] Boissonnat, J.D, Yvinec, M., *Algorithmic Geometry""*, Cambridge, UK, (1988).

[9] Jason. C, Zhiru. Z, "An Efficient and Versatile Scheduling Algorithm Based on SDC Formulation", *Proceedings DAC 2006*, San Francisco, California, USA, June 24-26, (2006).

[10] Elad H., Seshadhri C., "Efficient Learning Algorithms for Changing Environments", *Proceedings of the 26th International Conference on Machine learning*, Montreal, Canada, (2009).

[11] Hendrik B., Jan S., "Efficient Algorithms for Decision Tree Cross-Validation", *Journal of machine Learning Research*, Vol. 3, (2002), pp. 621-650.

[12] Ning Z., Zhixiong C., Guozhen X., "Efficient Elliptic Curve Scalar Multiplication Algorithms Resistant to Power Analysis", *Journal of Information Sciences*, Vol. 177, (2007), pp. 2119-2129.

برمجية مطورة للتصميم الهيدروليكي للمسقى المحسنة

محمد أحمد نصار * وهشام عبد المعز محمد عبد الجواد **

[*] قسم المياه ومنشآت المياه –كلية الهندسة بالزقازيق – جامعة الزقازيق، *nasserzagazig@yahoo.com*

[**] قسم الفيزيقا ولرياضيات الهندسية –كلية الهندسة بالمطرية – جامعة حلون، *habdelmoez@yahoo.com*

E م١١٠٢L١L٨ ¾,÷s1› 5¸e و ؛ م٠١٠٢L٧L٩٢ ¾,÷s1› م2eF

ملخص البحث. :م2ã: ozœ ãe,9א «,„aj,א9s «,„aj,א9s õ2¦2ç ¸ e õ,9g ¿a,í,i y :9ð¦,o ãzıi »,m„,›א ¸¦,Sא
@e2: «א,,maو ¿g ãð> ¿a ن93: «,3¸÷›א ozœو • ãQázzSא ـط9zQ›א «א ذ y,›א «,3¸u 9„QQ=›
«,gç و ¤ı=¿ çi ãðZא ¿a o,„Sא 5ãz› ã¦,و,g õ2>,ma «א22aو • ãáı=¿ ع א9í;i «,a,Qg و ã¦9í,fiو ã„m„¦,
ـط9zQ›א ç,m> ل5s ¿a ¿„i,íeא ـط9ðs ¾ @e2=›א „3„و,2„œ ¸ s,ş , ¿ãSא ¿a,íz›א و Kل,Q2=¥£א
ã„eא,zŞא «,a9ı2Sא 9ªí ð¦א,s 5¦9z=i ً,Q¦i م9ã¦ و • ã¦,gz›א «,a,Qg ¸ => ã3¸÷›א ـط,ãí ,„¾ ¾
ã„ı2á›א

¸ ı> ¿a,íz›א אzœ @„¸ð: y 2e و • ã„m„¦,›א ¿Q›א ãð> ãðãí »›zs و «,a,QQ›א و ¿„i,íeא ـط9ðs çi „Qe,
2ç9¦£ aíi م9ı2Sא ¿a و K,Qa ¾ ã„a9ã›א «,>و,÷Sא ¿a ?א2>א و 22¦ yz›א و EııP y,›א ,¦9ð: ع و,÷aF
«,e,Q: ـ9ç و 5ãa • õـ2> y¦,2ç م¿=ı¦ y,›א ,¦9ð: ع و,÷a ن e «,A9ı2Sא ozœ 5s ¿=z¦ ➢ و,2a ¿a,í,i
¿a õـ2> ¿„sא,: ـ9ç و • ع و,÷Sא @b,zS ـ2> ¸ ¦,a ¿zãa ـ9ç و • y,›א ã3¸÷› ãáı=zSא ع,eeא ¾ õـ2>
ç,ms »›ذ çi ãe,ge,i ¿a,íz›א م9ã¦ و אzœ Kz„áz=›א ـ9ã2› ,ã¸b ,gaא2z=¥א ¿u ½›א و «,zQSא
z„áz=›א ـ9ã> v,zi »›zs و • ã3¸÷›א vא ¿çi و ¿Q›א ãðZ ã¦ـ,Q=e£א ¿ íא9Şא »›ذ ¾ ,ç ã„›,¾eא ¤„›,3=›א
K»i9¥ 5ãa õ,9g÷a ã„3„› و,2„œ ¿aא,i ,a 5a,2=›א ã„í,3ai و • ã¦,QSא y,›א õ,א ز و «,ágא9a «,¸ıð=S ,ãe و
ª>9› 2e و ً,ã¸ma ً,¦ و 2¦ ãQQQSא ¸ e,mSא 92i ¸ ı> a:,¦,2a و • ½z¸›א ع 9g9a ¿a,íz›א אzœ ,,¸=sא y
2e و 92i õv,ás «,2e ,Qs • ً,¦ و 2¦ ,gi,m> y ½›א »›ı: و ¿a,íz›א ,ga2e ½›א ¿¦,=z›א ½i 2„ç @eא9:
ـ9ç و

Kã„i,m2א ç¿a9„Şא 2>א9e مא2z=¥,i «,„aj,א9”א