The WOWA operator and the interpolation function W^*: Chen and Otto's interpolation method revisited

Vicenç Torra
Dpt. d'enginyeria informàtica
Escola Tècnica Superior d'Enginyeria (ETSE)
Universitat Rovira i Virgili
Carretera de Salou, s/n
E-43006 Tarragona (Spain)
E-mail: vtorra@etse.urv.es

Abstract

The WOWA operator, a combination function that generalizes both the weighted mean and the OWA operator, is based on a interpolation function to calculate a new set of weights from two initial sets of weights. In this paper we study a variation of Chen and Otto interpolation method that is adequate to the WOWA operator. We report several errors that appear in that article and introduce some new limit conditions that fit our requirements.

1. Introduction

In these days, the need of combination functions in all fields (e.g., mathematics [12, 7], economy [19, 11], biology [5]) is increasing due to the fact that data is more easily obtained. Within Artificial Intelligence, they are used in several environments (e.g., robotics [3], vision [10], knowledge acquisition [15]) for two major purposes: when a system makes a decision or when it needs a comprehensive representation of the domain.

When the objects to synthesise are numeric values two combination functions are found in the literature. On one hand we find the weighted mean. This classical function, that has as a particular case the arithmetic mean, has been studied in [1, 2]. On the other hand, recently R. Yager [17, 18] introduced a new combination function named the OWA (Ordered Weighted Aggregation) operator that has already been used in several applications [19, 9]: The properties of this functions are studied in [8] and in [6].

Both functions, the weighted mean and the OWA operator are to combine values according to a set of weights. However, the meaning of these weights is different in both functions. The weighted mean computes a value that synthesises the ones of the information sources (experts or sensors) taking into account the reliability of these sources. This is, each value is weighted according to the reliability of its supplier. The OWA operator, instead, combines the information allowing to weight the values in relation to their ordering position. In this way, a system can diminish the importance of extreme values increasing the importance of central ones.

Therefore, in both combination functions there is a set of weights that determine the output. However, the meaning of weights in both combination functions is different. In order to take advantage of both sets of weights we introduced in [15] the WOWA (Weighted OWA) that combines the advantages of both combination functions. The new function allows the user to weight the reliability of the information source, as the weighted mean does, and the values in relation to their relative position, as the OWA operator.

Due to the fact that this combination function relies on an interpolation method, we study one of these methods in this paper. In particular, we study the one described in [4] . We show that their main guidelines are correct, but that the paper contains several errors. In this paper we correct the errors detected and introduce a variation of the algorithm in order to fit our requirements.

The structure of the paper is as follows. In section 2, we introduce the WOWA operator, and in section 3 we revisit the interpolation method introduced in [4]. The paper finishes with the conclusions.

2. The WOWA operator

In this section we introduce the definition of the WOWA operator and we give some considerations about its calculation. The properties of this new operator and its relation with the weighted mean and the OWA operator are studied in [14, 15]. In [14] this function has been generalized to the Quasi-WOWA and to the L-WOWA.

Definition 1. Let **p** and **w** be weighting vectors of dimension n ($\mathbf{p}=[p_1 \ldots\ p_n]$, $\mathbf{w}=[w_1, \ldots\ w_n]$) such that:

i) $p_i \in [0,1]$ and $\sum_i p_i = 1$

ii) $w_i \in [0,1]$ and $\sum_i w_i = 1$

In this case, a mapping $f_{wowa}: \mathbb{R}^n \rightarrow \mathbb{R}$ is a **Weighted Ordered Weighted Averaging (WOWA) operator** of dimension n if

$$f_{wowa}(a_1,\ldots,a_n) = \sum_i \omega_i\, a_{\sigma(i)}$$

where $\{\sigma(1),\ldots,\sigma(n)\}$ is a permutation of $\{1,..,n\}$ such that $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ for all i=2, ..., n. (i.e., $a_{\sigma(i)}$ is the i-th largest element in the collection $a_1,\ldots, a_n$), and the weight ω_i is defined as

$$\omega_i = W^*(\sum_{j \le i} p_{\sigma(j)}) - W^*(\sum_{j < i} p_{\sigma(j)}) \qquad (1)$$

with W^* a monotonic increasing function that interpolates the points (i/n, $\sum_{j \le i} w_j$) together with the point (0,0). W^* is required to be a straight line when the points can be interpolated in this way. This later condition is required to prove [14] idempotency (i.e., $f_{wowa}(a,...,a) = a$). From now on, ω represent the set of weights $\{\omega_i\}$, i.e., $\omega=[\omega_1,..., \omega_n]$.

Given a set of weights **p** and **w**, and the data vector **a**, to calculate f_{wowa} (**a**) we should calculate first the set S of points to interpolate. This set is defined as:

$$S=\{(i/n, \sum_{j \le i} w_j) \mid i=1,...,n\} \cup \{(0,0)\}$$

Then, we interpolate S with a function (say W^*). The synthesised value is obtained after ordering the input values a_i (i.e., determining the permutation σ such that $a_{\sigma(i-1)} \ge a_{\sigma(i)}$) and calculating $\sum_i \omega_i a_{\sigma(i)}$ with the weights $\omega = \{\omega_i\}$ defined according to the previous definition.

To obtain W^* from the weighting vector **w** we need an interpolation method. Among all interpolation methods the one to be used should define a monotonic and bounded function when input data is monotonic and bounded. As this is the case of the method introduced in [4] we have used it as a starting point. This method, based on the algorithm of McAllister and Roulier for producing a second-degree Bernstein polynomial, obtains from a set of monotonic or convex observations a monotonic or convex function. The method has to be slightly modified to satisfy the statement in definition 1 that W^* is required to be a straight line when the points can be interpolated in this way. According to the authors claim in [4] the method obtains an interpolation curve with a null derivative at the extreme points d_1 and d_n. Therefore, the interpolation method can not obtain a straight line although the points are of this form.

In the definition of the WOWA operator, another approach is also possible. It consists on defining the function W^* without considering the initial step of defining the weight vector ω. In this way, we can use as W^* any monotonic increasing function within the [0,1] interval satisfying $W^*(0)=0$ and $W^*(1)=1$. In fact, this idea is also present in [18], as Yager defines a way to extract the weights from fuzzy quantifiers when they are *regular monotonically non-decreasing* (i.e., they satisfy $Q(0)=0$, $Q(1)=1$ and if $x>y$ then $Q(x) \ge Q(y)$, conditions that the weighting function W^* should satisfy). Recently, Yager [16] has defined a OWA operator that considers the importance of the elements. In this case, the weights are calculated from a quantifier as in (1).

3. Chen and Otto interpolation method revisited

According to definition 1 in section 2, the WOWA operator needs of an interpolation method to build function W from weights **w** and **p**. Recall that the interpolation function has to be monotonic increasing and with the initial point (0,0) and the final point (1,1). Besides of that, the function has to be a line when the points can be interpolated in this way.

In this section we review the interpolation method described in [4] to define membership functions. We correct several errors in their paper and we show that their assumption of null derivative in the extreme point is not satisfied by their method. We introduce a new method for defining extreme derivatives that is more adequate in our case and that achieves a smooth function in the extremes.

As the interpolation method is based on the second-degree Bernstein polynomial, we begin with this definition and proving a proposition that is used later on.

Definition 2 [4]. Let $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be arbitrary points and let w=(a,b) be an arbitrary point with $a=(x_1+x_2)/2$. Let g be the piecewise linear spline passing through the points P_1 and P_2 and w with a single discontinuity at a. The second-degree Bernstein polynomial $B_2[P_1, w, P_2]$ of g on $[x_i,x_2]$ is defined as:

$$B_2[\mathbf{P}_1,\mathbf{w},\mathbf{P}_2](x) = B_2(g)(x) = \frac{g(x_1)\,(x_2-x)^2 + 2\,b\,(x - x_1)\,(x_2 - x) + g(x_2)\,(x - x_1)^2}{(x_2 - x_1)^2}$$

The second-degree Bernstein polynomial satisfies the following proposition.

Proposition 1. Let $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be two arbitrary points and let w=(a,b) be an arbitrary point with $a=(x_1+x_2)/2$ and with $b=(y_1+y_2)/2$. In these conditions, the Bernstein polynomial is a line.

From now on, it is asumed that the set of points to be interpolated are $\{d_1=(x_1,\mu_1),..., d_n=(x_n,\mu_n)\}$. This is, each d_i is of the form (x_i, μ_i). It is assumed that $x_i<x_{i+1}$ for all i=1,..., n-1. Therefore, the interpolation function $\mu(x)$ is going to be defined in the interval $[x_1, x_n]$.

The interpolation procedure of Chen and Otto follows the following structure:

1. For each point d_i, its slope m_i is calculated.
2. A knot point o_i is inserted between each d_i and d_{i+1} and a second-degree Bernstein polynomial is defined between contiguous pairs of points.

This procedure follows the guidelines introduced in [4]:

1. About m_i:

(a) m_i must be consistent with the monotonicity and convexity of the piecewise linear function determined by the data points d_{i-1}, d_i and d_{i+1}.

(b) m_i must vary continuosly with respect to changes in s_i and s_{i+1} (s_i is defined below as $s_i=(\mu_i - \mu_{i-1}) / (x_i - x_{i-1})$).

(c) Points that are maximum or minimum points are fixed to have an slope equal to zero (i.e., $m_i=0$ for all points d_i such that $\mu_i>\max(\mu_{i-1}, \mu_{i+1})$).

(d) Extreme points d_1 and d_n are also required to have an slope equal to zero (i.e., $m_1=m_n=0$).

2. About knot points:

(e) Only one knot point should be required between two data points. This is to minimize the complexity of the algorithm.

We review next the interpolation method. We present directly the improved version, and we comment at the end of each step the corrections introduced.

3.1. The interpolation algorithm

1. Definition of slope m_i for each point d_i.

First we define s_i as:

$$s_i=(\mu_i - \mu_{i-1}) / (x_i - x_{i-1})$$

and then m_i is calculated according the following set of cases:

(i) If $s_i\ s_{i+1} \le 0$, then $m_i=0$. This garantees that maximum and minimum points are assigned slopes 0. This also segments the entire data into monotonically nonincreasing or nondecreasing subsets.

(ii) If $|s_i| > |s_{i+1}| > 0$ and $s_i\ s_{i+1} > 0$, we extend a line through d_i with slope s_i until it intersects the horizontal line through d_{i+1} at the point $b=(b_x, \mu_{i+1})$. We then define m_i as the slope of the line that goes through d_i and the medium point of b and d_{i+1}. This is

$$m_i = (\mu_{i+1} - \mu_i) / (c_x - x_i) \text{ where } c_x = (b_x + x_{i+1}) / 2$$

(iii) If $0 < |s_i| < |s_{i+1}|$ and $s_i\ s_{i+1} > 0$, we reverse the above procedure by extending the line through d_i with slope s_{i+1} until it intersects the horizontal line through (x_{i-1}, μ_{i-1}) at the point (b_x,μ_{i-1}). Then we set $c_x = (b_x + x_{i-1}) / 2$ and $m_i = (\mu_i - \mu_{i-1}) / (x_i - c_x)$.

(iv) If $s_i = s_{i+1}$, then $m_i = s_i$

(v) The end-point slopes m_1 and m_n are set to zero explicitly since this is required by guideline (b).

In relation to [4], we have corrected several misspelling errors, and included in (ii) and (iii) that $s_i\ s_{i+1}$ should be greater than zero (to make these cases non overlapping with (i)). Besides of that we have added condition (iv) as conditions in [4] were not exhaustive (the definition of m_i in this case, coincides with the value that would be obtained in (ii) or (iii)).

2. Knot point insertion and definition of the interpolation curve.

Following [4], we assume nondecreasing data points (for nonincreasing data points the same algorithm can be applied without change). This is the case of interest in relation to the WOWA operator.

Let R_i be the rectangle determined by the points d_i and d_{i+1}. Let L_i be the line that passes through (x_i, μ_i) with slope m_i. There are three distict cases regarding the intersection of the neighboring slope lines L_i and L_{i+1}, depending on whether the knots change the local convexity of the spline or not.

(i) L_i and L_{i+1} intersect at a point $z=(t_i, z_i)$ in R_i and z is neither d_i nor d_{i+1}. Let

$$v_i = ((x_i + t_i)/2, L_i((x_i+t_i)/2))$$
$$w_i = ((x_{i+1} + t_i) / 2, L_{i+1} ((x_{i+1} + t_i) / 2)$$

and let L be the line joining v_i and w_i, then the knot point o_i is defined as follows:

$$o_i = (t_i, L(t_i))$$

In this case, the interpolation function $\mu(x)$ is defined on $[x_i, x_{i+1}]$ as:

$$\mu(x) = \begin{cases} B_2 [d_i, v_i, o_i] (x) \text{ on } [x_i,t_i] \\ B_2 [o_i, w_i, d_{i+1}] (x) \text{ on } [t_i, x_{i+1}] \end{cases}$$

(ii) L_i and L_{i+1} do not intersect within R_i or their intersection is d_i or d_{i+1}. The knot o_i is determined similarly as in case (i), i.e., $o_i=(t_i, L(t_i))$, but with $t_i = (x_i + x_{i+1}) / 2$. The definitions of v_i, w_i, μ_i remains as in case (i) and this is also the case for $\mu(x)$.

(iii) L_i coincide with L_{i+1}. In this case we interpolate the function with a line. This definition is equivalent to apply case (ii) because in this latter case proposition 1 is satisfied because $L_i(t_i) = (\mu_i+\mu_{i+1})/2$.

In relation to [4] and besides of some misspelling changes (e.g., their definition of v_i and w_i was not correct) we have inserted case (iii) because cases (i) and (ii) are not exhaustive.

Notice that in the definition of $\mu(x)$ the limits of the rectangle R_i should be considered as not in R_i. d_i and d_{i+1} should be considered out of R_i because if we assume that they are in R_i, then case (i) is applied and then condition (d) above is not satisfied. Notice that the intersection of L_1 and L_2 is in some cases d_1 (always that $m_1=s_1$). In this case if (i) is applied then $z=d_1$ and $v_1=d_1$ and $w_1 = ((x_1+x_2)/2, L_2((x_1+x_2)/2))$. Therefore $\mu(x)$ is defined in $[x_1, x_2]$ by the following Bernstein polynomials

$$B_2[d_1, d_1, d_1](x) \text{ on } [x_1, x_1]$$
$$B_2[d_1, w_1, d_2](x) \text{ on } [x_1, x_2]$$

Notice that the former is a "*punctual*" polynomial, and that the latter is a line because proposition 1 holds. Therefore in this case condition (d) is not satisfied.

The requirement that d_i and d_{i+1} should be considered out of R_i is not stated in [4] and therefore the procedure is not correct. Besides of that, figure 13 in that paper is neither correct. The authors show an example of a non-differentiable membership function and three interpolation curves when 3, 6 and 9 points of the original membership function are considered to apply the interpolation method. It can be seen that in one case, the one corresponding to the function built from 3 points, m_1 and m_n has a value of zero while in the other two cases the value is not zero. These results contradicts authors assessments.

3.2. Slope and extreme points d_1 and d_n.

With the procedure described in the previous section, we have corrected the definition of Chen and Otto [4]. However, as it was said in the beginning of section 3, this is not adequate in our case because we should interpolate a straight line when points are of this form. This contradicts the definition of the interpolated function as it is forced that m_1=m_n=0. Below we introduce a new method of defining m_1 and m_2 that solves this drawback and is general enough to obtain smooth functions in any case.

The new definition of end-point slopes to replace (v) in point 1 of section 3.1 is as follows:

Definition of m_1 and m_n.

(i) $m_1 = s_2^2 / m_2$

if $m_2 = 0$ and $s_2 = 0$ then $m_1 = 0$

(ii) $m_n = s_n^2 / m_{n-1}$

if $m_{n-1} = 0$ and $s_n = 0$ then $m_n = 0$

To analyse this definition, we should consider the different cases that follows from point 1 in section 3.1. We only consider the cases (i) to (iv) because (v) is the one that we are redefining now. We have decomposed case (i) in two cases according to the value of s_i. As we have assumed nondecreasing points, s_i can be greater or equal than zero:

(i.a) $s_i>0$ and $s_i\ s_{i+1} \le 0$. In this case as m_i is forced to be zero $s_i > m_i = 0$.
(i.b) $s_i=0$ and $s_i\ s_{i+1} \le 0$. In this case as m_i is forced to be zero $s_i = m_i = 0$.
(ii) $|s_i| > |s_{i+1}| > 0$ and $s_i\ s_{i+1} > 0$. In this case $m_i < s_i$.
(iii) $0 < |\ s_i\ | < |\ s_{i+1}\ |$ and $s_i\ s_{i+1} > 0$ (notice that in this case $s_i \neq 0$) In this case $m_i > s_i$.
(iv) $s_i = s_{i+1}$. In this case $m_i = s_i$.

From these cases, when i=2 we can study the definition of m_1. It can be seen that to obtain a smooth function in d_2 the following conditions should hold. The proof of these conditions is given in brakets.

(i.a) m_1 should tend to infinity (see figure 1).

(i.b) m_1 should be zero (proof: from definition - see figure 2).

(ii) m_1 should satisfy $m_1 > s_2$ (proof: as $0<m_2<s_2$, then $0 < 1 < s_2/m_2$ and $0< s_2 < s_2^2/m_2$ - see figure 3).

(iii) m_1 should satisfy $0 < m_1 < s_2$ (proof: as $m_2 > s_2 > 0$, then $1 > s_2 / m_2 > 0$ and $s_2 > s_2^2/m_2 > 0$ - see figure 4).

(iv) m_1 should be s_2 (proof: from definition $m_1 = s_2^2 / m_2 = s_2$ - see figure 5).

Also from the case given above, we can study the definition of m_n. As before we study the conditions that lead to a smooth function. The proofs of these conditions are in brakets.

(i.a) According to the value of s_n (see figure 6) there are two alternatives.

If $s_n = 0$, then m_n should be zero (proof: from definition, as $m_{n-1} = 0$ and $s_n = 0$).

If $s_n<0$, then m_n should tend to minus infinity

(proof: This also holds as $s_n<0$ and $m_{n-1}=0$).

(i.b) The value of m_n depends on the value of s_n (see figure 7).

If $s_n > 0$, then m_n should tend to infinity.

If $s_n = 0$, then m_n should be zero.

If $s_n < 0$, then m_n should tend to minus infinity.

(proof: all conditions follow from $m_{n-1}=0$ and the value of s_n).

(ii) m_n should satisfy $0 < m_n < s_n$ (proof: As $m_{n-1} > s_n > 0$, then $1>s_n /m_{n-1}>0$ and $s_n>s_n^2 /m_{n-1} > 0$ - see figure 8).

(iii) m_n should satisfy $s_n<m_n$ (proof: as $s_n>m_{n-1}$ then $s_n /m_{n-1} > 1$ and $s_n^2/m_{n-1}>s_n$ - see fig 9).

(iv) m_n should be s_n (proof: as $m_{n-1} = s_n = s_{n-1}$, then by definition $m_n = s_n^2 / m_{n-1}$ that is s_n - see figure 10).

3.3. An example

In figures 11 and 12 we show the interpolating curbes corresponding to the points {(0,0), (.5,.2), (1,1)}. The former figure corresponds to the revised method of Chen and Otto, and the latter to the method introduced here.

Conclusions

The paper revisites the interpolation method of Chen and Otto. We show and correct some of the mistakes detected in their method and introduce a new definition of the slopes corresponding to the end-points. This new definition is to fit the requirements of the combination function that we have introduced in [13, 14].

References

[1] Aczél, J., *A short course on functional equations*, (D. Reidel Publishing Company, 1987).
[2] Aczél, J., Alsina, C., On synthesis of Judgements, Socio-Econom Plann. Sci, (1986).
[3] Amat, J., Esteva, F., López de Mántaras, R., Autonomous Navigation Troup for Cooperative Modelling of Unknown Environments, *Proceedings of the Int. Conference on Advanced Robotics* (ICAR'95), Sant Feliu de Guixols, Catalunya, Spain (1995) 383- 389.
[4] Chen, J.E., Otto, K.N., Constructing membership functions using interpolation and measurement theory, *Fuzzy Sets and Systems*, 73:3 (1995) 313-327.
[5] Day, W.H.E., McMorris, F.R., Critical comparison of consensus methods for molecular sequences, *Nucleic Acids Research*, 20:5 (1992) 1093-1099.
[6] Fodor, J., Marichal, J.-L., Roubens, M., Characterization of the Ordered Weighted Averaging Operators, *IEEE Trans. on Fuzzy Systems*, 3:2 (1995) 236-240.
[7] Genest, C., Zidek, J.V., Combining Probability Distributions: A Critique and An Annotated Bibliography, *Statistical Science*, 1:1 (1986) 114-148.
[8] Grabisch, M., Fuzzy integral in multicriteria decision making, *Fuzzy Sets and Systems* 69 (1995) 279-298.
[9] Herrera, F., Herrera-Viedma, E., Verdegay, J.L., A model of consensus in group decision making under linguistic assessments, *Fuzzy Sets and Systems*, 78 (1996) 73-87.
[10] López, B., Álvarez, S., Millán, P., Puig, D., Riaño, D., Torra, V., Multistage vision system for road lane markings and obstacle detection, Proceedings of *Euriscon '94*, pp. 489-497, Malaga (1994).
[11] Raiffa, H., *Decision Analysis: Introduction Lectures on Choices under uncertainty*, (Addison-Wesley Publishing Company, 1968).
[12] Roberts, F. S., On the indicator function of the plurality function, *Mathematical Social Sciences*, 22 (1991) 163-174.
[13] Torra, V., The Weighted OWA operator for synthesis of information, *Fifth IEEE International Conference on Fuzzy Systems (IEEE-FUZZ'96)*, September 8-11, 1996, New Orleans, USA. (accepted).
[14] Torra, V., The Weighted OWA operator, *Int. J. of Int. Systems* (accepted for publication).
[15] Torra, V., Cortés, U., Towards an automatic consensus generator tool: EGAC, *IEEE Transactions on Systems, Man and Cybernetics*, 25:5 (1995) 888-894.
[16] Yager, R. R., Quantifier Guided Aggregation Using OWA operators, *International Journal of Intelligent Systems*, 11 (1996) 49-73.
[17] Yager, R.R., On ordered weighted averaging aggregation operators in multi-criteria decision making, *IEEE Trans. on SMC*, 18 (1988) 183-190.
[18] Yager, R.R., Families of OWA operators, *Fuzzy Sets and Systems*, 59 (1993) 125-148.
[19] Yager, R.R., Goldstein, L.S., Mendels, E., FUZMAR: An approach to aggregating market research data based on fuzzy reasoning, *Fuzzy sets and systems*, 68 (1994) 1-11.

Figures:

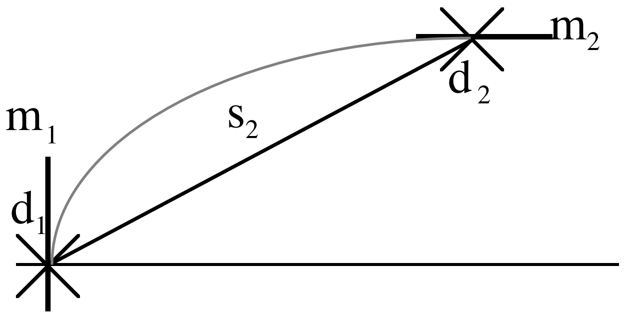

Figure 1

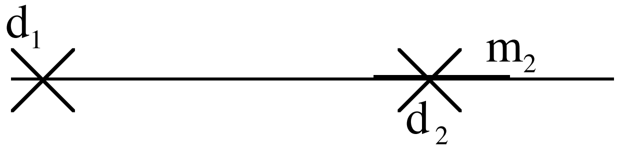

Figure 2

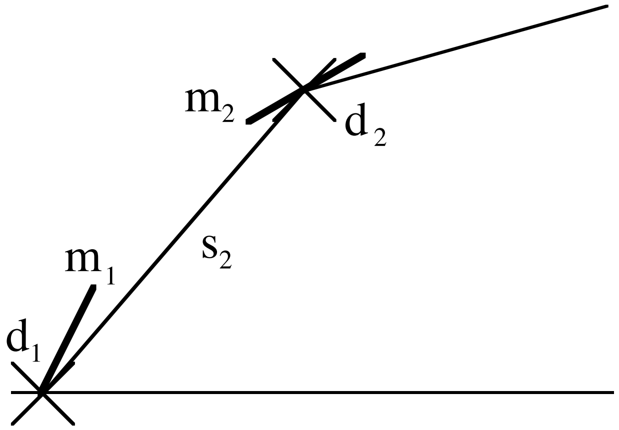

Figure 3

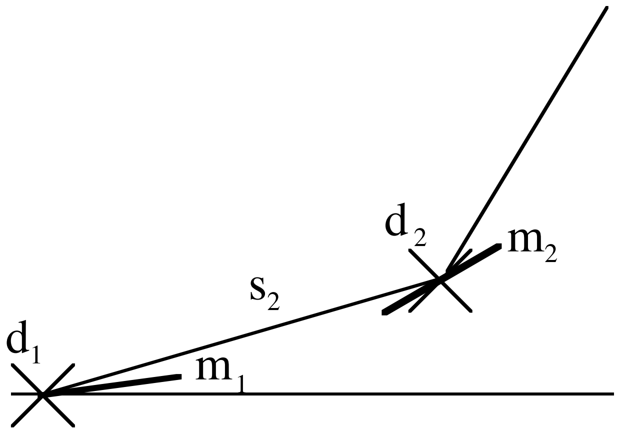

Figure 4

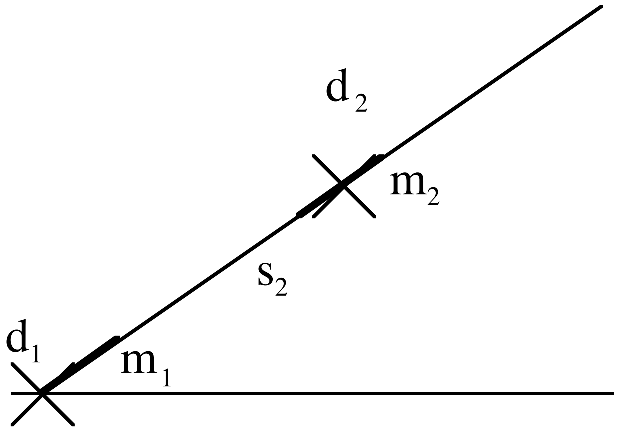

Figure 5

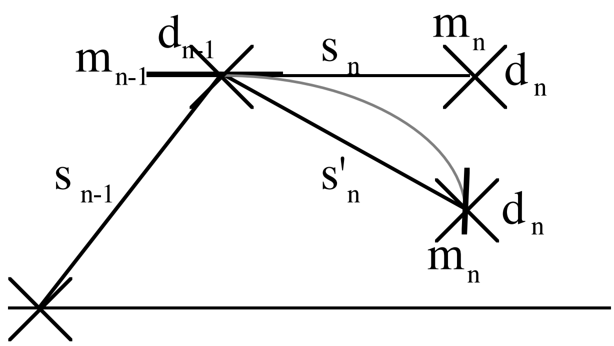

Figure 6

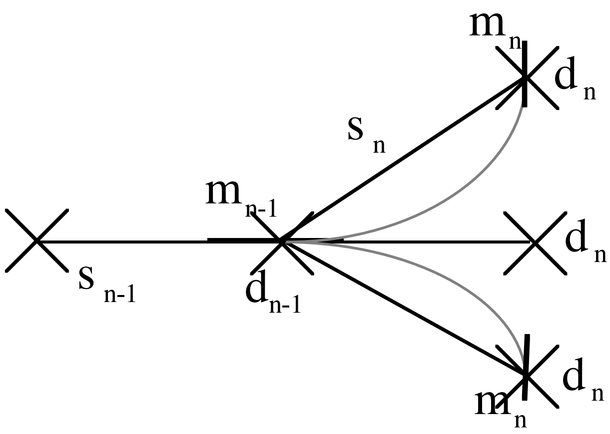

Figure 7

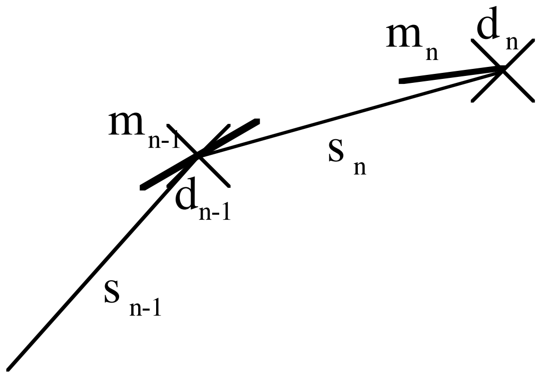

Figure 8

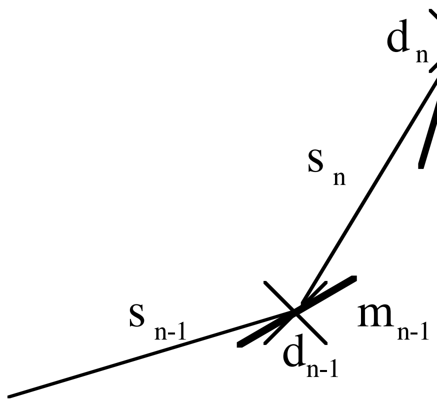

Figure 9

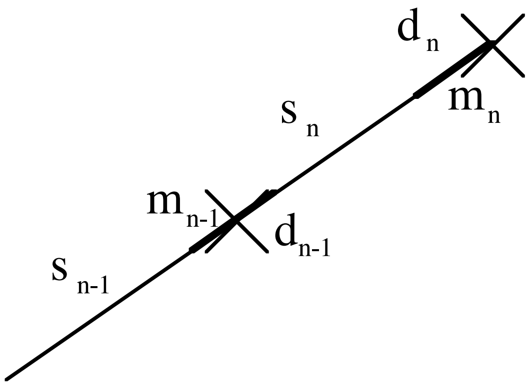

Figure 10

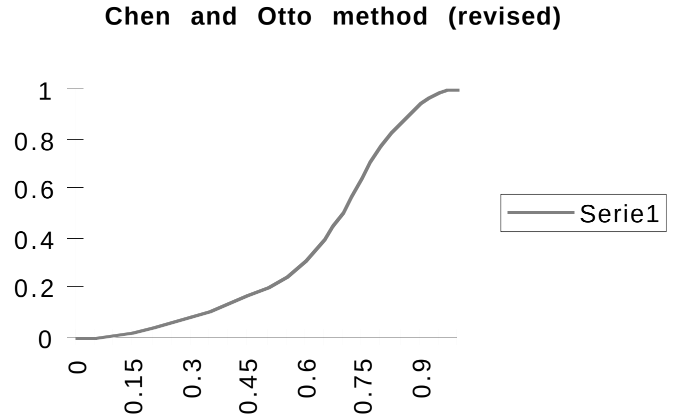

Figure 11

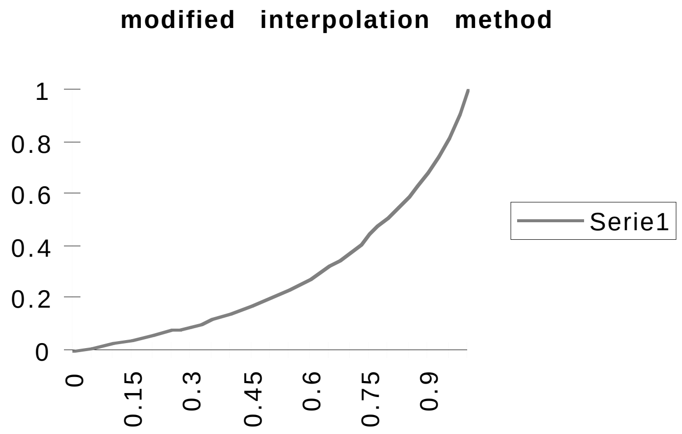

Figure 12