Finding Transition Points
Zero-Crossings, Maxima, Minima and Inflections

Kenneth Baclawski
Northeastern University

Abstract

A function has a transition point when some property changes at the transition point. Such points represent important features of a function. We examine three kinds of transition points: sign, monotonicity and convexity. A sign transition point, usually called a zero-crossing, is a place where the function changes from negative to positive or vice versa. At a monotonicity transition point the function changes from increasing to decreasing or vice versa. At a convexity transition point the function changes from being convex to concave or vice versa. When the function is continuous these three kinds of transition points are roots, maxima/minima, and inflection points, respectively. In this note we specify algorithms for finding the three kinds of transition points. We formally prove that the algorithms always converge, and we compare these algorithms with other algorithms. In the proof of the algorithm for inflections, we give a geometric interpretation of the mediant and its connection to convexity.

Introduction

A TRANSITION POINT is a place where a function has an abrupt change of some property. Formally, a function $f(x)$ has a transition point for properties P and Q at $x = x_0$ if there are real numbers $a < x_0$ and $b > x_0$ such that f has property P on the open interval (a, x_0) and has property Q on the open interval (x_0, b). Note that the function $f(x)$ is not assumed to be continuous, and the intervals where f has the properties P and Q do not include x_0[1].

We will consider three kinds of transition point as follows:

Sign transition point. The sign of the function changes from negative to positive or vice versa. Such a point is called a zero-crossing. When the function is continuous, the point is a root (zero) of the function.

Monotonicity transition point. The function changes from increasing

[1]Technically, the transition point is the point $(x_0, f(x_0))$ in the plane. For simplicity we will omit the reference to $f(x_0)$ and refer to x_0 as being the transition point.

to decreasing or vice versa. If the function is continuous, such a transition point is a local extreme point (i.e., a maximum or minimum).

Convexity transition point, The function changes from convex to concave or vice versa. If the function is continuous, the transition point is an inflection point.

Transition points are often confused with other kinds of points. For example, zero-crossings are commonly confused with roots. However, these two notions are different in general. Consider the following function:

$$g(x) = \begin{cases} 1/x & \text{if } x \neq 0, \\ 1, & \text{if } x = 0. \end{cases}$$

The function $g(x)$ is shown on the left in Figure 1. This function has no roots but has a zero-crossing at $x = 0$. Next consider the following function:

$$h(x) = \begin{cases} 1/x^2 & \text{if } x \neq 0, \\ 0, & \text{if } x = 0. \end{cases}$$

The function $h(x)$ is shown on the right in Figure 1. This function has a root at $x = 0$ but has no zero-crossings.

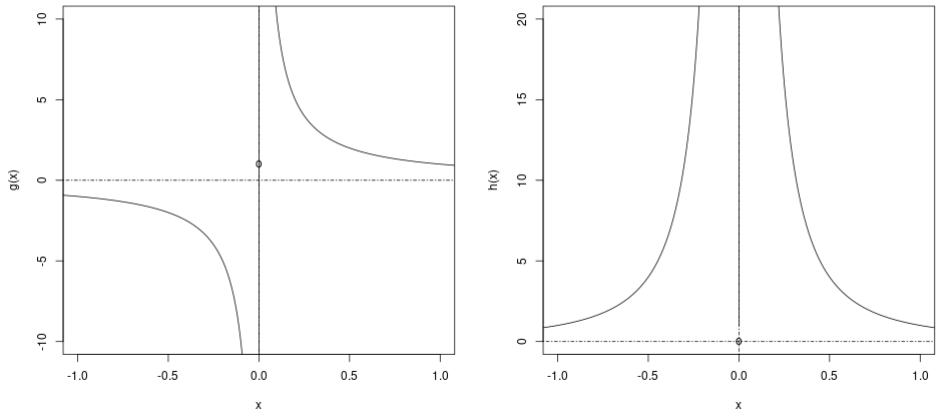

Figure 1: Left: A function that has a zero-crossing but no roots
Right: A function that has a root but no zero-crossings

The distinction between a root and a zero-crossing is significant since an algorithm for finding a zero-crossing would not be as effective in general for finding roots and vice versa. For example, the bisection method is often described as a method for finding a root of a continuous function when it is actually a method for finding a zero-crossing of a function that need not be continuous.

As another example, consider the following function:

$$k(x) = \begin{cases} x^2(1+\sin^2(1/x)) & \text{if } x \neq 0, \\ 0, & \text{if } x = 0. \end{cases}$$

The function $k(x)$ is shown on the left in Figure 2. This function has a minimum at $x = 0$, which is not a monotonicity transition point. On the other hand, consider the following function:

$$l(x) = \begin{cases} 1/x^2 & \text{if } x \neq 0, \\ 10, & \text{if } x = 0. \end{cases}$$

The function $l(x)$ is shown on the right in Figure 2. This function has a monotonicity transition point but has no minima or maxima.

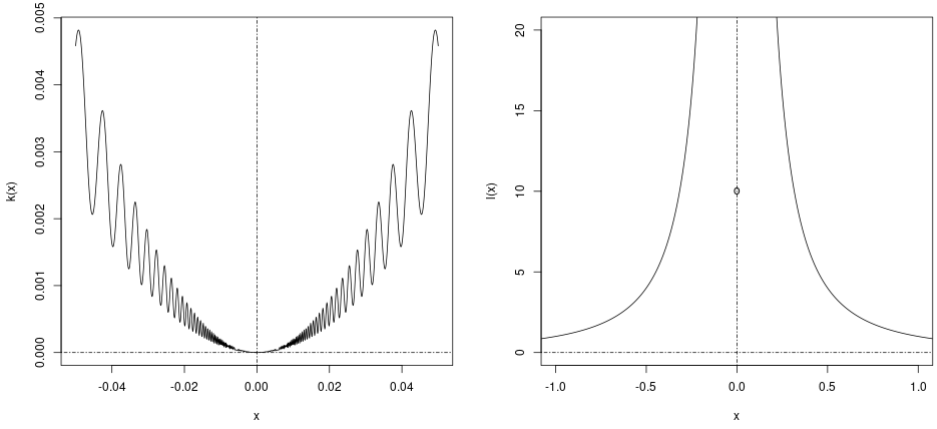

Figure 2: Left: A minimum point that is not a monotonicity transition point Right: A monotonicity transition point that is not a minimum or maximum

In any algorithm for finding a special point, the function and possibly derived functions will need to be evaluated. As computations will generally have limited precision, computed numbers can be indistinguishable from one another even though the theoretical numbers are different. In this paper we will write η for the accuracy of the computed function and any derived expressions. For two numbers y and z, we write $y \approx z$ to mean $|y - z| \leq \eta$, i.e., y and z are indistinguishable up to the specified precision.

We begin by describing the well-known bisection method for finding a zero-crossing. We then define bisection-style algorithms for finding monotonicity and convexity transition points. Each of the bisection-style algorithms are compared with other methods for finding special points of functions. In the Appendix we prove that the newly introduced algorithms are correct and therefore converge. One interesting feature of the proof of Theorem 2 is its use of the mediant for numbers other than integers. We also give a geometric interpretation of the mediant that visually shows its connection with convexity.

Zero-Crossings

There are number of methods for finding a zero-crossing. All of the methods start with an interval such that the function has opposite signs on the endpoints, and the methods iteratively reduce the size of the interval, which confines the zero-crossing to smaller and smaller intervals. The method terminates when the interval is smaller than a specified length or the function value is equal to zero within the precision of the computation of the function. This forces the zero-crossing to be within the final interval, up to the precision of the computations. Algorithms with the property that the desired point is always within an interval at each iteration are called *bracketing methods*. The simplest bracketing method is the bisection method, which simply splits the interval exactly in two at each iteration. Another bracketing method is the *false position* method, also called the *regula falsi* method. This method uses the intersection of the secant rather than the midpoint. The Interpolate Truncate and Project (ITP) method is an improvement of the false position method that adjusts the secant intersection in a number of ways that are determined by three parameters (Oliveira and Takahashi, 2020).

The performance of an algorithm for finding a zero-crossing is measured by the number of evaluations of the function. The bisection method

is optimal in the sense that its worst case performance is at least as good as any other algorithm for finding a zero-crossing (Sikorski, 1982). The bisection algorithm has the additional advantage that it is simple and requires very little additional computation beyond a function evaluation in each iteration.

While the bisection method has optimal worst case performance, the other bracketing methods can have much better performance on some classes of functions. Moreover, the ITP method is claimed to have the same optimal worst-case performance as the bisection method (Oliveira and Takahashi, 2020). However, it is worth noting that part of the reason why the ITP method achieves the optimal worst-case performance of the bisection method is that it defaults to the bisection method when it is unable to improve on the bisection method. Indeed, every algorithm for finding a zero-crossing can also achieve the same optimal worst-case performance as the bisection method by defaulting to the bisection method.

Every bracketing method requires one to specify the function f and the interval (a, b) where one is looking for a zero-crossing. The interval must be within the domain of the function, and the function must have opposite signs on the two endpoints of the interval. One must also specify the desired accuracy $\epsilon > 0$ of the zero-crossing point and the computational precision $\eta > 0$.

The are two cases for opposite signs. We refer to each one as the "pattern" of the zero-crossing that is to be found. The following is the method written in pseudo code:

Iteration:

1. Interpolation: Compute a point $c \in (a, b)$ according to the particular algorithm. For the bisection algorithm, $c = \frac{a+b}{2}$. For the false position algorithm, $c = \frac{af(b) - bf(a)}{f(b) - f(a)}$,
2. Search for Pattern: At least one of the pairs (a, c), (c, b) will satisfy the required pattern.
3. Replace the Pair: Select the pair that satisfies the required pattern and replace (a, b) with the selected pair.

Termination Conditions: There are two conditions for terminating the algorithm.

1. If $b - a \leq 2\epsilon$, then the midpoint $\frac{a+b}{2}$ will be within ϵ of the zero-crossing. This can be checked at the beginning of the iteration.
2. if $f(c) \approx 0$, then c is a zero-crossing up to the computational precision. This can be checked after c has been computed. The accuracy is $b-a$. Continuing the algorithm would not be useful since the function is too close to being a constant equal to 0 for the computations to distinguish the values from being 0.

Each step of the bisection algorithm requires a function evaluation and a small number of arithmetic operations. The length of the interval is reduced by a factor of 2 at each step, so the algorithm will always terminate in no more than $\left\lceil \log_2 \frac{(b-a)}{2\epsilon} \right\rceil$ iterations. Furthermore, if a function has a unique zero-crossing in the interior of the initial interval, then the bisection method will converge to it because every interval during the iteration will contain the zero-crossing in its interior. As discussed above, the other bracketing algorithms for finding zero-crossings can have better performance, but will have the same worst-case performance as the bisection algorithm if they default to the bisection algorithm.

There are other methods that can be used for finding zero-crossings. The secant method uses the same formula as the false position method, but does not check the sign of the function on the newly computed point and uses the newly computed point together with the most recently computed point for the next iteration. The secant method is not a bracketing method, and it can fail to converge to the zero-crossing.

Newton's method is a well-known algorithm for finding a root of a continuously differentiable function (Newton's Method, n.d.). However, it has a number of disadvantages compared with the bracketing methods. One must compute and implement the derivative of the function, so it is necessary to know the analytic formula for the function. Furthermore, Newton's method can converge slowly or even diverge, although one can mitigate this problem by monitoring the performance of Newton's method and defaulting to the bisection method when convergence is too slow. On the other hand, Newton's method can achieve high rates of convergence. For example, it can achieve quadratic convergence (i.e., the number of places in the approximation increases quadratically or better) when the function is twice continuously differentiable.

Monotonicity Transition Points

An extreme point (extremum) is one at which a function has a maximum or minimum. A frequently used method for finding an extremum is to differentiate the function and then find a root of the derivative. This is not quite accurate. A root of the derivative need not be an extremum. For example, the function $f(x) = x^3$ has derivative $f'(x) = 3x^2$ which is 0 at $x = 0$ but $f(x)$ does not have an extremum at $x = 0$. All one can say in general is that if $f(x)$ is differentiable and it has an extremum at x, then the derivative is 0 at x. The converse is not true. On the other hand, a function can have an extremum without being differentiable or even continuous at the extremum, as in the case of the function $l(x)$ shown in Figure 2.

We now define a method for finding a monotonicity transition point of a function, which we call the Bisection Monotonicity Method. The method has the advantage that it does not require one to differentiate the function. As with the ordinary bisection method above, there will be two patterns depending on whether one is seeking a minimum or a maximum. The algorithm proceeds as follows:

Input: Function f; three numbers a, b and c in the domain of f such that $a < b < c$ and such that either $f(a) \leq f(b) \geq f(c)$ (for finding a maximum) or $f(a) \geq f(b) \leq f(c)$ (for finding a minimum); a desired accuracy $\epsilon > 0$; and a computational precision $\eta > 0$.

Iteration:

1. Bisection: Compute the midpoints $d = \frac{a+b}{2}$ and $e = \frac{b+c}{2}$.
2. Search for Pattern: At least one of the triples (a, d, b), (d, b, e) and (b, e, c) will satisfy the required pattern. The proof that this search always succeeds is given in Theorem 1 of the Appendix.
3. Replace the Triple: Select any one of the triples that satisfy the required pattern and replace (a, b, c) with the selected triple.

Termination Conditions: There are two conditions for terminating the algorithm.

1. If $c - a \leq 2\epsilon$, then the midpoint $\frac{a+c}{2}$ will be within ϵ of the extremum.
2. if $f(a) \approx f(b) \approx f(c)$, then the function is constant in the interval (a, c) up to the specified precision, so no further processing can distinguish any of the points in the interval (a, c).

For both of the termination conditions, the midpoint $\frac{a+c}{2}$ will be within $\frac{c-a}{2}$ of the extremum. Both termination conditions can be checked at the beginning of the iteration.

In the Iteration step one could first try one of the midpoints, say d, and if $f(d)$ is greater than or equal to $f(a)$ and $f(b)$, then the pattern has been found without computing the other midpoint and its value. Accordingly, each step of the Bisection Monotonicity Method requires one or two function evaluations and a small number of arithmetic operations. Note that the initial triple (a, b, c) need not be equally spaced. However, if the initial triple is equally spaced, then all subsequent triples during the iteration will also be equally spaced. In this case, the length of the interval is reduced by a factor of 2 at each step, so the algorithm will terminate in $\left\lceil \log_2 \frac{(c-a)}{2\epsilon} \right\rceil$ iterations.

Furthermore, if f is a function on an interval $[a, b]$, and if $m \in (a, b)$ has the property that f is strictly increasing on $[a, m]$ and strictly decreasing on $[m, b]$ (or vice versa), then the Bisection Monotonicity Method converges to m. To see why this is so, first note that when a function is strictly increasing or decreasing, no triple will satisfy the pattern. So no triple during the iteration can be entirely within either $[a, m]$ or $[m, b]$. Therefore, every interval in the iteration contains m in its interior, and it follows that the sequence of intervals converges to m.

Convexity Transition Points

An inflection is a point at which a function changes from being concave to convex or vice versa. A frequently used method for finding an infection is to differentiate the function twice and then find a root of the second derivative. This is not quite accurate. A root of the second derivative need not be an inflection. For example, the function $f(x) = x^4$ has second derivative $f''(x) = 12x^2$ which is zero at $x = 0$, but $f(x)$ does not have an inflection at $x = 0$. All one can say in general is that if $f(x)$ is twice differentiable and it has an inflection at x, then the second derivative is 0 at x. The converse is not true.

We now define a method for finding a convexity transition point of a function, which we call the Bisection Convexity Method. The method has the advantage that it does not require one to differentiate the function. As with the ordinary bisection method above, there will be two patterns.

The algorithm proceeds as follows:

Input: Function f; four numbers $a < b < c < d$ in the domain of f such that either $\frac{f(b)-f(a)}{b-a} \leq \frac{f(c)-f(b)}{c-b} \geq \frac{f(d)-f(c)}{d-c}$ or $\frac{f(b)-f(a)}{b-a} \geq \frac{f(c)-f(b)}{c-b} \leq \frac{f(d)-f(c)}{d-c}$; a required accuracy $\epsilon > 0$; and a computational precision $\eta > 0$.

Iteration:

1. Bisection: Compute the midpoints $p = \frac{a+b}{2}$,$q = \frac{b+c}{2}$, and $r = \frac{c+d}{2}$.
2. Search for Pattern: At least one of the quadruples (a, p, b, q), (p, b, q, c), (b, q, c, r) and (q, c, r, d) will satisfy the required pattern. The proof that this search always succeeds is given in Theorem 2 of the Appendix.
3. Replace the quadruple: Select any one of the quadruples that satisfy the required pattern and replace (a, b, c, d) with the selected quadruple.

Termination Conditions: There are two conditions for terminating the algorithm.

1. If $d - a \leq 2\epsilon$, then the midpoint $\frac{a+d}{2}$ will be within ϵ of the convexity transition point.
2. If $\frac{f(b)-f(a)}{b-a} \approx \frac{f(c)-f(b)}{c-b} \approx \frac{f(d)-f(c)}{d-c}$, then the function is linear in the interval (a, d) up to the specified precision, so further processing will no distinguish any convexity properties in (a, d).

For both of the termination conditions, the midpoint $\frac{a+d}{2}$ will be within $\frac{d-a}{2}$ of the extremum. Both termination conditions can be checked at the beginning of the iteration.

In the Iteration step one could first try two of the midpoints, say p and q, and if one of the quadruples (a, p, b, q) and (p, b, q, c) satisfies the pattern, then it is not necessary to compute the last midpoint and its value. Accordingly, each step of the Bisection Convexity Method requires two or three function evaluations and a small number of arithmetic operations. If the initial quadruple (a, b, c, d) is equally spaced, then all quadruples in the algorithm will also be equally spaced. As a result, the denominators in a pattern will all be the same, and it will not be necessary to perform the divisions. In this case, the length of the overall interval is reduced

by a factor of 2 at each step, so the algorithm will always terminate in $\left\lceil \log_2 \frac{(d-a)}{2\epsilon} \right\rceil$ iterations.

Furthermore, if f is a function on an interval $[a,b]$, and if $q \in (a,b)$ has the property that f is strictly convex on $[a,q]$ and strictly concave on $[q,b]$ (or reversing convex and concave), then the Bisection Convexity Method converges to q. To see why this is so, first note that when a function is strictly convex (respectively, concave), the slopes between successive points are strictly increasing (respectively, strictly decreasing), and hence do not satisfy the required pattern. So no quadruple during the iteration can be entirely within either $[a,q]$ or $[q,b]$. Therefore, every interval in the iteration contains q in its interior, and the sequence of intervals converges to q.

Conclusion

We have introduced the transition point, a new kind of special point for real-valued functions of a single variable that can be useful for analyzing properties of such functions. We have also introduced bracketing algorithms for finding extreme points and inflections using bisections. The new algorithms have a number of advantages compared with other techniques, such as being guaranteed to converge and not requiring any additional effort such as differentiating the function one or more times. The only significant disadvantage is that the algorithms do not have as good a performance as other techniques. Furthermore, when using the other techniques, it can be useful to default to the newly introduced bracketing algorithms so that one has both higher performance for many cases and guaranteed convergence in all cases.

References

Mediant (n.d.). Wikipedia page on the mediant. http://bit.ly/3KWSyDQ

Newton's Method (n.d.). Wikipedia page on Newton's method. http://bit.ly/3ZJDDkS

Oliveira, I. F. D.; Takahashi, R. H. C. (2020). An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality. *ACM Transactions on Mathematical Software*. 47 (1): 5:1–5:24. doi:10.1145/3423597. ISSN 0098-3500.

Sikorski, K. (1982). Bisection is optimal. *Numerische Mathematik*. 40 (1): 111–117. doi:10.1007/BF01459080. ISSN 0945-3245. S2CID 119952605.

Appendix

In this Appendix, we provide the proofs that the iterations of the algorithms introduced in this paper always succeed, and therefore the algorithms always converge. We first prove that the Search for Pattern step of the Bisection Monotonicity Algorithm always succeeds for the maximum pattern.

Theorem 1 *Let f be a function on the interval $[a_1, a_5]$, $a_3 \in (a_1, a_5)$, $a_2 \in (a_1, a_3)$ and $a_4 \in (a_3, a_5)$. If $f(a_1) \leq f(a_3) \geq f(a_5)$, then there exists $i \in \{1,2,3\}$ such that the inequality pattern $f(a_i) \leq f(a_{i+1}) \geq f(a_{i+2})$ holds.*

Proof. There are four cases depending on how $f(a_3)$ compares with $f(a_2)$ and $f(a_4)$.

1. $f(a_3) \leq f(a_2)$ and $f(a_3) \leq f(a_4)$. Since $f(a_1) \leq f(a_3)$, it follows that $f(a_1) \leq f(a_2)$. Therefore, $f(a_1) \leq f(a_2) \geq f(a_3)$ and the pattern with $i = 1$ holds.
2. $f(a_3) \leq f(a_2)$ and $f(a_3) > f(a_4)$. This implies $f(a_1) \leq f(a_2) \geq f(a_3)$ as in the first case above.
3. $f(a_3) > f(a_2)$ and $f(a_3) \leq f(a_4)$. Since $f(a_3) \geq f(a_5)$, it follows that $f(a_4) \geq f(a_5)$. Therefore $f(a_3) \leq f(a_4) \geq f(a_5)$ and the pattern with $i = 3$ holds.
4. $f(a_3) > f(a_2)$ and $f(a_3) > f(a_4)$. These two imply the pattern with $i = 2$.

Note that in the first case above, both cases $i = 1$ and $i = 3$ hold. In every case the pattern holds for at least one i and the theorem follows.

Reversing all of the inequalities in Theorem 1 above gives the result for the minimum pattern.

We now prove that the Search for Pattern step of the Bisection Convexity Algorithm always succeeds.

Theorem 2 *Let f be a function on the interval $[a_1, a_7]$. Partition $[a_1, a_7]$ into 6 subintervals with the sequence $\{a_2, a_3, a_4, a_5, a_6\}$. If $\frac{f(a_3)-f(a_1)}{a_3-a_1} \leq \frac{f(a_5)-f(a_3)}{a_5-a_3} \geq \frac{f(a_7)-f(a_5)}{a_7-a_5}$, then there exists $i \in \{1,2,3,4\}$ such that the inequality pattern $\frac{f(a_{i+1})-f(a_i)}{a_{i+1}-a_i} \leq \frac{f(a_{i+2})-f(a_{i+1})}{a_{i+2}-a_{i+1}} \geq \frac{f(a_{i+3})-f(a_{i+2})}{a_{i+3}-a_{i+2}}$ holds.*

The proof of the theorem will use the following operation:

The *mediant* of ratios a/c and b/d, where $cd > 0$ is the ratio $\frac{a+b}{c+d}$ (Mediant, n.d.). Although the mediant is only defined for ratios of integers, it is obviously meaningful for ratios of real numbers in general. The property of the mediant that we will use is the following proposition. This proposition is easily proved by multiplying by the denominators and by canceling the common term. There is a proof in Mediant (n.d.),

Proposition 1 *If a, b, c and d are real numbers such that $cd > 0$ and $\frac{a}{c} < \frac{b}{d}$ (respectively, $\frac{a}{c} \leq \frac{b}{d}$), then $\frac{a}{c} < \frac{a+b}{c+d} < \frac{b}{d}$ (respectively, $\frac{a}{c} \leq \frac{a+b}{c+d} \leq \frac{b}{d}$).*

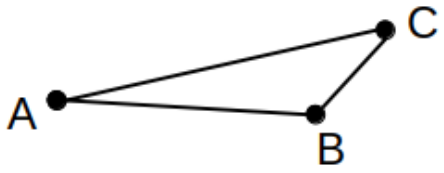

Figure 3: A geometric interpretation of the mediant

The mediant has a geometric interpretation. Suppose that one has three points A, B and C that proceed from left to right in the plane, i.e., the x-coordinates of A, B and C are strictly increasing. Figure 3 shows an example. Then the slope of AC is the mediant of the slopes of AB and BC.[2] Geometrically, it is easy to see that the slope of AC will always be between the slope of AB and the slope of BC, which is the conclusion of Proposition 1. Moreover, ABC will be concave or convex depending on the order of the slopes of AB and BC.

[2]To be more precise, the ratio defining the slope of AC is the mediant of the ratios that define the slopes of AB and BC.

Proof of Theorem 2. For simplicity in the proof, we will write y_j for $f(a_j)$. There are 32 cases depending on how $\frac{y_{j+1}-y_j}{a_{j+1}-a_j}$ compares with $\frac{y_{j+2}-y_{j+1}}{a_{j+2}-a_{j+1}}$, for $j \in \{1,2,3,4,5\}$. For technical reasons, the comparisons for $j = 1,2$ are $>$ and $\leq$; whereas the comparisons for $j > 2$ are $\geq$ and $<$. To deal with the large number of cases, we group them into sets denoted by five inequality symbols or asterisks. For example, the set $\{>>\geq **\}$ denotes the set of cases in which

$$\frac{y_2-y_1}{a_2-a_1} > \frac{y_3-y_2}{a_3-a_2}, \frac{y_3-y_2}{a_3-a_2} > \frac{y_4-y_3}{a_4-a_3}, \frac{y_4-y_3}{a_4-a_3} \geq \frac{y_5-y_4}{a_5-a_4},$$

and the remaining two inequalities can be in either direction, so the set $\{>>\geq **\}$ has 4 cases in all.

Set 1. $\{>>\geq **\}$ Applying Proposition 1 to each of the first three inequalities gives the following inequalities:

$$\frac{y_2-y_1}{a_2-a_1} > \frac{y_3-y_1}{a_3-a_1} > \frac{y_3-y_2}{a_3-a_2} > \frac{y_4-y_2}{a_4-a_2} > \frac{y_4-y_3}{a_4-a_3} \geq \frac{y_5-y_3}{a_5-a_3} \geq \frac{y_5-y_4}{a_5-a_4}$$

In particular, we have that $\frac{y_3-y_1}{a_3-a_1} > \frac{y_5-y_3}{a_5-a_3}$. However, by hypothesis we have that $\frac{y_3-y_1}{a_3-a_1} \leq \frac{y_5-y_3}{a_5-a_3}$. It follows that none of the 4 cases in Set 1 ever occur.

Set 2. $\{** <<<\}$ Proceed as in Set 1 by applying Proposition 1 three times. As with Set 1, it follows that none of the 4 cases in Set 2 ever occur.

Set 3. $\{\leq> ***\}$ The first two inequalities imply the pattern with $j = 1$. There are 8 cases in this set, one of which is in Set 2.

Set 4. $\{* \leq\geq **\}$ The second and third inequalities imply the pattern with $j = 2$. There are 8 cases in this set, none of which are in earlier sets.

Set 5. $\{** <\geq *\}$ The third and fourth inequalities imply the pattern with $j = 3$. There are 8 cases in this set, two of which are also in Set 3.

Set 6. $\{*** <\geq\}$ The last two inequalities imply the pattern with $j = 4$. There are 8 cases in this set, five of which are in previous sets.

All 32 cases have been accounted for, and for every case that can occur, one or more of the inequality patterns hold. The theorem then follows.

Reversing all of the inequalities in Theorem 2 above gives the result for the other inflection pattern.

KENNETH BACLAWSKI is an Associate Professor Emeritus at the College of Computer and Information Science, Northeastern University. Professor Baclawski does research in data semantics, formal methods for software engineering and software modeling, data mining in biology and medicine, semantic collaboration tools, situation awareness, information fusion, self-aware and self-adaptive systems, and wireless communication. He is a member of the Washington Academy of Sciences, IEEE, ACM, IAOA, and is the chair of the Board of Trustees of the Ontolog Forum.