Automatic analysis of D-partition

V G Bogaevskaya
PhD student, Yaroslavl State University, Yaroslavl, Russia
BogaevskayaVG@yandex.ru

Abstract. The paper is dedicated to automatization of D-partition analysis. D-partition is one of the most common methods for determination of solution stability in systems with time-delayed feedback control and its dependency on values of control parameters. A transition from analytical form of D-partition to plain graph has been investigated. An algorithm of graph faces determination and calculation of count of characteristic equation roots with positive real part for appropriate area of D-partition has been developed. The algorithm keeps an information about analytical formulas for edges of faces. It allows to make further analytical research based on the results of computer analysis.

1. Introduction
Mathematical modeling in spheres such as physics, biology, neurodynamic and many others often tends to necessity to research the solution dynamic for differential equations with parameters. Among thousands of examples a special place belongs to differential equation with parameters. Starting with 1992 when K. Pyragas published his article about the control of chaos with help of time-delayed feedback [1], this field became more and more popular due to easy physical implementation and high effectivity [2, 3]. For this type of problems it is important to determine solution stability. Usually for this aim researchers use the method of D-partition suggested by Yu. I. Neimark [4]. In accordance with Lyapunov theorem, if we linearize given system on the periodic solution or equilibrium state and construct characteristic equation, the count of its roots with positive real part will give us an information about the solution stability in most cases. The method of D-partition reflects imaginary axis of roots plane to the plane of two chosen parameters, dividing it to areas with constant number of roots from right complex half-plane and invariable solution stability. An analysis of D-partition consists in a calculation of these numbers for all areas in special order. Usually researchers do this process manually, but there are systems for which such analysis is very complicated and time-consuming because of complex structure of the D-partition [5]. Also for systems with big number of parameters it may be needed to analyze huge amount of significantly different D-partitions. All of this is the reason why the issue of process automatization is an actual problem.

2. Statement of the problem

2.1. General statement
Consider some differential or difference-differential equation with two real parameters α and β. Let linearize it on the periodic solution or equilibrium state. For linearized system it is possible to construct characteristic equation
To investigate the issue of solution stability we need to know the number of roots of the equation (1) that have positive real part. We suppose that \( \mu \) continuously depends on parameters, so the only possible situation, when the number of roots from right complex half plane can change, is crossing of imaginary axis during parameters changing.

We suppose that from analytical research of (1) we know:

1) Formulas for D-curves on plane of parameters \((\alpha, \beta)\) that give us
   a) a root of characteristic equation moving through the point of origin when \( \beta \) is changing  
   \[ \alpha = l_i(\beta), \]
   b) a pair of roots moving through complex conjugate points \( \pm i\omega \) on imaginary axis when \( \beta \) is changing  
   \[ \alpha = a_i(\omega), \quad \beta = b_i(\omega). \]
2) Formulas of conditions that determine the direction of roots movement through imaginary axis
3) Number of roots with positive real part for the case when \( \alpha = \beta = 0. \)

Also we define a rectangle on the plane of parameters \((\alpha, \beta)\) that contains the area for computer analysis. We will name it plot area.

In this paper we pose the problem to analyze automatically the D-partition defined by information given above and to calculate the count of roots with positive real parts for each area inside given plot area.

2.2. The statement of problem in terms of graphs
The problem can be reformulated in terms of graphs. Let consider following points as vertexes of graph:

1) Points of crossing axes and borders of the plot area with each other
2) Points of crossing D-curves with axes and borders
3) Points of crossing D-curves with each other and itself
4) Points on D-curves where the condition of movement direction changes its state

\[ h(\mu, \alpha, \beta) = 0, \]  
(1)

Figure 1. Types of vertexes

Notice that one point can satisfy to several criteria.

Let consider parts of D-curves, axes and borders connecting two vertex as edges. Due to choice of vertexes, it is obvious that

1) Edges cross each other only in vertexes and these vertexes are ends for the given edges,
2) Roots movement direction is the same for all points of each edge.

As a result the graph faces will have following properties:

1) Each face belongs to only one area of given D-partition,
2) Each area of D-partition may include several graph faces, but if two faces are adjacent, their common edge always will be part of axis. So, the whole area can be constructed by union of faces that have such type of common edge.
3) Due to using axes as graph edges, it’s always possible to determine two edges for each of four faces around the point of origin. It’s very important because the analysis of D-partition should be started with one of them.

So, in these terms, the main problem is to construct the graph, based on analytically gained formulas, to find all its faces and to calculate the count of characteristic equation roots with positive real part for each of them in special order.

3. The algorithm of graph’s elements determination

3.1. Search of vertexes
Let approximate each D-curve with polygon. For this aim we set value of step h. For the case of movement through the point of origin we will change $\beta$ from its minimum to its maximum value on the plot area. For the case of pair of roots movement through points $\pm i\omega$ we will use h to change $\omega$ from zero to some given value. As a result we will be able to operate by analytically gained D-curves using a computer.

3.1.1. Search of points of crossing for two curves. By definition the search of most vertexes types requires a method of finding crossing points for two curves. We use the standard formulas for crossing of two segments to determine an existence of the point of curves crossing. There are examples demonstrating that crossing of segments isn’t equivalent to curves crossing, but we suppose that h is small enough to give the segment that approximates this part of curve with high accuracy on every step. If the point of crossing segments exists, we divide the increment (of $\beta$ for the movement through the point of origin case and $\omega$ for the other one) by two and calculate appropriate points on curves. As a result we have four segments and we check crossing for them. If we find the pair of crossing segments, we use for them the same algorithm as described above. The process stops when the length of segments becomes less than admissible error. Furthermore, we keep an information which formula was used for each edge, therefore, it gives us an opportunity to precise points of crossing analytically.

3.1.2. Crossing of borders and axes. There are three types of border crossing: self-crossing of borders, crossing with axes and crossing with D-curves. For the first two cases we know coordinates of crossing points and can include them to the list of vertexes at the beginning without any calculations. To find points of crossing D-curves with borders, on each step we check whether a new point on D-curve is inside the plot area or not. If the state of this condition changes, we start the process of crossing point search for each of borders and the current part of the D-curve. Noteworthy that every D-curve should have its start and end outside of the plot area, on the border or in crossing point with some other D-curve, otherwise we will have dangling vertex but by the definition of D-partition, it can’t have any of them. The D-curve of movement through the point of origin starts and ends with values of $\beta$ equal to the minimum and maximum values for $\beta$ inside the plot area accordingly, so there will not be such problems here. The D-curve of movement through $\pm i\omega$ starts with $\omega = 0$. It is easy to see that this point will lay on D-curve of movement through the point of origin. To make computer analysis finite, we need to restrict the maximum value of $\omega$, but in this case we can’t guarantee where this point will be. We use the indicator of being inside the plot area to avoid the situation with dangling vertex. If maximum value of $\omega$ is reached but the current point is inside plot area we will continue increasing $\omega$ until new point leaves the plot area or stay at the same place during huge amount of steps. In the first case we find the point of crossing with border, in the second one the point should lay on some other D-curve. It’s possible to prove that then it coincides with the point of direction of movement changing on the second D-curve or the edge based on it and previous vertex can be excluded from consideration. So, we mark such points as vertex to analyze it later.
If points of border crossing have been found, we create an array of subsegments that lay inside the plot area and use them for further analysis. Points of axes crossing could be found similarly. We check, in which coordinate quarter current point lays, and if the value changes, start the process of crossing point search. When the point of crossing is found we check if we already have a vertex distance, to which from this point is smaller than admissible error. If we find such vertex, we take it as the current crossing point and add an information about the current D-curve and value of changing variable to it. If we haven’t such vertex, we add a new one to the list of them.

3.1.3. Self-crossing of D-curves. Let divide plot area by a grid with relatively small cell width. As it was mentioned above increasing changing variable we move in the D-curve. After completion of each step we keep an information which cells of the grid are crossed by the current part of D-curve. So, when we process a new segment (or subsegment, if we have found points of crossing with borders and/or axes) we determine in which cells it lays and check crossing only with previously processed segments that also are there. If we find a point of crossing, we check whether this point was already added as a vertex and either add a new vertex, or update an information about D-curves in the old one. Also we replace stored segment by two with ends in new crossing point. We split current segment to two subsegments too and continue analysis with them separately. As a result all stored segments will have vertexes only as ends. It will be useful for edges search.

3.1.4. Direction of movement changing. At the end of analysis of the segment or subsegment we check the state of movement direction condition for its end points. If states are different, we precise the point by dividing the changing variable by two and check which of new subsegments has different states of the condition at its ends. We repeat the process until the length of subsegment becomes smaller than admissible error and then we add a new vertex to the list.

3.2. Edges searching
When all vertexes are found we can define edges of the graph. For each D-curve we store a list of vertex sorted by changing variable. We go through this list and set edges as parts of D-curve that connect two neighboring vertexes. However some edges can lay outside of the plot area and we should exclude them from analysis. Sufficient condition of exclusion is that an arbitrary point from the edge is out of the plot area because we already found all border crossing points and they are vertexes in our graph. So, before adding edge to the list we check if it is inside the plot area or not. Also for each edge we calculate a polar angle in each of its vertexes. We take this value as the angle of the small segment approximating the curve near the vertex. We also add this information to the vertex, where we sort list of its edges by the angle.

4. An algorithm of graph faces searching and classification
In accordance with the rules of D-partition analysis our algorithm should work in following order:
1) The first face that we find should have the point of origin as its vertex. Only for such faces we know the count of roots with positive real part initially.
2) When some faces are already found and the number of roots is calculated for them, we should choose as the next face the one that is adjacent with already processed face. To make it easier to detect such faces, we will count for each edge how many processed faces it belongs. So, if the face has an edge with the number equal to 1 and this edge isn’t a border (borders can be “used” in faces only once), we can process it.
3) We will repeat step 2 until there will be no edge with the index “1”. But we still can have edges with the index “0” if we have nested faces. In this case we need to determine inside which face nested component located, find its outer contour, mark them by “1” and then go to step 2.

Let consider the process in more details.
4.1. Faces around the point of origin

Due to using axes as edges we can easily get four vertexes nearest to the point of the origin and belonging to one of them. Let choose two such vertex: one from positive part of $\beta$-axis and one from positive part of $\alpha$-axis. Let name the edge connecting the vertex on $\beta$-axis with the vertex in the point of the origin “start edge”. In the list of edges of the vertex from the point of the origin, sorted by polar angle, the second edge is a neighbor of the start edge. Besides we know that it is on anticlockwise direction. We keep this information and start following process: among all edges from the list of the vertex on $\alpha$-axis we choose a neighbor of given edge that lay on the same direction as we store (anticlockwise in this case). This new edge also belongs to the face we determine. Then we do the same thing but for the second end of new edge and continue this process until we get the start edge as the next one. As a result we have full list of edges defining the face. By the problem statement the number of roots of characteristic equation with positive real part is already known. We increase the index for all these edges to 1 and store the information about found face to the each of edges. We can do the same process for other three faces, but it’s not necessary because they will be processed anyway as adjacent faces.

![Figure 2](image.png)

Figure 2. Search of faces. Left picture is the search of the face around the point of origin. Right one is the search of adjacent face. Numbers show the index of edge. Arrows are the direction of movement. Grey field is already processed face.

4.2. Search of adjacent faces

Let take one of the edges with index “1” as the start edge. Consider one of its two vertexes and its list of edges. If there is only one other edge, we take it as the next one and look at its second vertex. If there are several other edges, we choose the neighbor by polar angle, that doesn’t belong to the processed face which contains start edge. We keep an information about the direction of movement to the neighbor from the current edge (clockwise or anticlockwise) and continue round of the face according to it until we get the start edge as the next one. As before we increase the index for all these edges to 1 and store the information about found face to the each of edges.

4.3. Calculation of root count for the adjacent face

From the statement of the problem we know how the count of roots will change if we cross an edge in the direction of $\beta$ increasing. If $\beta$ decreases, the count will change in opposite way. Therefore, to calculate how many roots have positive real part if parameters $\alpha$ and $\beta$ are taken from the inside of processing face we need to detect which face is above of common edge and which one is below. Let hold a vertical line through an arbitrary point on the start edge and count the number of its crossing with edges of the processing face. It has been proved that if this number is even, the processing face is located below the start edge, but if the number is odd, the processing face is located above. In the first case we need to deduct the value, got from the condition, from the count of roots for adjacent face processed earlier. In the second case we will add this value.

4.4. The case of nested components

The process described above allows to find faces only for one coherent component of the graph. If we processed all edges with index “1” but we still have edges with index “0”, consider one of the edges never used before and start the search of nested component. Let take an arbitrary point on the edge and
hold a vertical line through it. Consider all points of crossing this line with edges and find the nearest one that lays above the point and belongs to the edge with index “2”. For this edge we know its faces, their numbers of roots and the condition of changing roots count when $\beta$ grows up. So it’s possible to get which face is located below. It is proved that this face contains given nested component. Consider the last edge with index “0” crossed by the line and located above the point. This edge belongs to outer contour of the nested component. Using it find all edges of outer contour using standard algorithm, set their indexes to “1” and start the same process as before.

5. Summary
The algorithm described in this paper allows us to construct D-partition based on analytically gained formulas inside specified rectangle, find all its areas as lists of edges and calculate the count of characteristic equation roots with positive real part for each of them. It also keeps an information about formulas used for each edge. It gives more accurate calculations and can be useful, if we want to continue analytical research after computer processing. Furthermore, algorithm can automatically determine a possibility of stabilization by the check whether face with the count of roots equal to zero exists or not.

6. References

[1] Pyragas K 1992 Continuous control of chaos by self-controlling feedback. Phys. Lett. A 170 421–28
[2] Flunkert V 2011 Delay-Coupled Complex Systems and Applications to Lasers (Berlin: Springer)
[3] Bogaevskaya V and Kashchenko I 2014 Influence of delayed feedback control on the stability of periodic orbits Automatic Control and Computer Sciences 48 53–65
[4] Neimark Yu I 1992 Robust stability and D-partition Automation and Remote Control 53 957–65
[5] Bogaevskaya V and Kashchenko I 2015 Cycle stabilization by one and two delay feedback control. Nonlinear Phenomena in Complex Systems 18 175–80