Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Control Engineering Practice 11 (2003) 493–504
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

System analysis via performance maps

J.J. Alpigini*, D.W. Russell

Engineering Division, Penn State Great Valley School of Graduate Professional Studies, 30 East Swedesford Road, Malvern, PA, 19355-1443, USA

Received 2 March 2000; accepted 17 June 2002

Abstract

While there are a number of visual methods common to the design and analysis of dynamic systems, they tend to be specific to their application and limited in the amount of information that they yield. This paper explores a visualization technique, titled the *performance map*, which is derived from the Julia set commonly used in the visualization of iterative chaos. Performance maps are generated via digital computation, and require a minimum of *a priori* knowledge of the system under evaluation. By the use of color-coding, these images convey a wealth of information to the informed user about dynamic behaviors of a system that may be hidden from all but the expert analyst.

Keywords: Visualization; Chaos; Linear control; Dynamics; Simulations

1. Introduction

There are a number of graphical techniques used in the classical design and analysis of dynamic systems, including time graphs, phase portraits, the root locus method, Bode diagrams and Nyquist plots. These are powerful analytic tools, yet each is constrained by its own set of limitations and tends to be rigidly focused, with limited utility. Because their graphical output is intentionally simple, each has a definite limitation in the amount of information that can be conveyed in a single image. And finally, each can require further mathematical analysis, which may be difficult for ill-defined or highly nonlinear systems.

This paper presents a method, termed the *performance map*, designed to visualize the behavior of a system's dynamic performance across variations in system parameters. This technique is derived from the familiar Julia set, which is used to visualize the occurrence of chaos in iterative mathematical formulae. The resulting visualization is color-coded, providing an immediate wealth of information to the knowledgeable viewer. Furthermore, because the performance map is developed algorithmically, via digital computation, its generation requires only moderate *a priori* knowledge and mathematical analysis.

This paper begins with a brief overview of the Julia set technique, in which the rules employed to color the image pixels is emphasized. The scheme is then adapted to fit the analysis of control and other dynamical systems by parameter mapping and appropriate rule selection. To demonstrate the effectiveness and veracity of the performance map, a simulation of a readily verified linear feedback system is considered. The generated map is demonstrated to accurately display multiple effects of parameter values on the system solution. The paper concludes with a performance map analysis of a complex system.

2. Julia sets

The performance map technique described in this paper is ultimately derived from the notion of Julia sets. Whilst the performance map is applicable to the visualization of control and other dynamic systems, Julia sets are used solely to provide color-coded descriptions of the motions exhibited by complex-valued iterative mathematical functions (Julia, 1918). The Julia set is the boundary between the set of points of a parametric function $Q_c(z)$ whose orbits escape towards infinity and the set of points whose orbits are attracted

*Corresponding author. Tel.: 1-610-648-3200; fax: 1-610-889-1334.
E-mail address: jja7@psu.edu (J. J. Alpigini).

0967-0661/02/$ - see front matter
PII: S0967-0661(02)00145-4

to some periodic cycle (Devaney, 1990). In this section the fundamental steps in generating a Julia set are reviewed, with particular emphasis on pixel mapping and evaluation rule generation. The reader already familiar with such sets can proceed to Section 3.

Julia set generation begins with pixel mapping and the selection of a color scheme. Each pixel ($p_{x,y}$) is mapped to a unique value of seed z_0 on the dynamic plane of $Q_c(z)$, where both variable z and parameter c are complex valued ($z = x + j \cdot y, c = a + j \cdot b$). The value of x is usually mapped to the horizontal axis and y is mapped to the vertical axis. The value of parameter c is held constant throughout the generation of any particular set.

The pixels are individually colored by rule evaluation following some arbitrary iteration period. For each pixel, i.e. seed value z_0, the equation is iterated for some preset maximum count. As the orbit progresses, selected values, e.g. magnitude, are evaluated against some "rule" to determine if the orbit is destined to approach infinity. Should the rule determine this to be the case, the orbit is said to have "escaped", and the iteration loop is exited immediately. Otherwise, the iteration is allowed to continue up to the fixed maximum iteration count. The pixel is colored according to how many iteration steps occurred before the iteration was halted, thus providing a measure of how long it took for the orbit to escape toward infinity, if at all. Any desired color scheme can be employed, but it is common to use black for an orbit that remained bounded for the full iteration period, red for an orbit that escaped in just one or two iterations, and other colors to represent orbits that escaped between these two extremes.

2.1. An example of the Julia rule

The rule employed to evaluate the escape potential of an individual orbit is unique to each function under study. Consider the simple iterative function, $Q_c(z) = z_{i+1} = z_i^2 + c$, where $z = x + j \cdot y$ and $c = a + j \cdot b$. For the purposes of digital computation, it is necessary to calculate the real and imaginary components of the function as shown in Eqs. (1) and (2):

$$Real: \quad x_{i+1} = x_i^2 - y_i^2 + a, \tag{1}$$

$$Imaginary: \quad y_{I+1} = 2 \cdot x_i \cdot y_i + b. \tag{2}$$

Devaney (1990) showed that an orbit of this system has escaped and is beginning to approach infinity if ever the real or imaginary component of z exceeds 2.0. A test of this boundary can be formalized as a Julia rule (Russell & Alpigini, 1997a, b):

Julia rule for $z_{i+1} = z_i^2 + c$

Rule trigger	$(\|x\| > 2.0)$ OR $(\|y\| > 2.0)$
Condition for iteration loop exit	Julia_rule_fired OR Maximum_iterations_exceeded

The rule is then employed with Algorithm 1 to color the individual pixel that maps to the particular values of 'x' and 'y', because parameter 'c' is to be fixed during the iteration process.

Algorithm 1.
Iteration loop for $z_{i+1} = z_i + c$, with Julia rule test

Variables	i: iteration counter x_i: real component of initial condition, z_i y_i: imaginary component of initial condition, z_i a: real component of parameter c b: imaginary component of parameter c
Initial conditions	$i = 0$ *Exit_Flag* = FALSE *Rule_Fired* = FALSE
Iteration loop	BEGIN Iteration_Loop $x_{i+1} = x_i^2 - y_i^2 + a$ {*calculate* $z_i + 1$} $y_{i+1} = 2 \cdot x_i \cdot y_i + b$ IF $(\|x_{i+1}\| > 2.0)$ OR $(\|y_{i+1}\| > 2.0)$ THEN *Rule_Fired* = TRUE ENDIF $i = i + 1$ {*increment iteration counter*} IF (*Rule_Fired*) OR ($i >$ *Maximum_Iterations*) THEN *Exit_Flag* = TRUE ENDIF LOOP UNTIL *Exit_Flag*
Final Action	Set_Pixel_Color(i)

2.2. Example of a Julia set

Fig. 1 shows the Julia set generated for $z_{i+1} = z_i^2 + c$, for parameter $c = 0.36 + j \cdot 0.1$, displayed next to the color key employed. Because the Julia set is the boundary between the sets of orbits which escape and those that remain bounded, the black area of the image, representing the bounded orbits, is termed the *filled-in* Julia set.

It is immediately apparent from an inspection of Fig. 1 that the function $z_{i+1} = z_i^2 + c$ is not a simple function at all. Rather, this seemingly benign equation exhibits surprisingly complex aspects that become apparent only when visualized in a Julia set.

3. Performance maps

It has been suggested that the Julia set method can be adapted to visualize control and other dynamic systems (Alpigini, 2000, 2002). Such a technique addresses the problem of visualizing automatically the state values of some system as its controller, or other, parameters are varied. In effect, the technique produces a color-coded *performance map* that reflects the dynamic performance of the system across intervals of system parameters. As with the Julia set, a performance map is generated via digital computation, using rules appropriate to the system for which dynamic behavior is being visualized.