ALGORITHMIC FINANCE

Stochastic flow diagrams

Neil J. Calkin; Marcos López de Prado

Algorithmic Finance (2014), 3:1-2, 21-42
DOI: 10.3233/AF-140032

Abstract, HTML, and PDF:
http://algorithmicfinance.org/3-1-2/pp21-42

Subscription, submission, and other info:
www.iospress.nl/journal/algorithmic-finance

Algorithmic Finance 3 (2014) 21–42
DOI 10.3233/AF-140032
IOS Press

Stochastic flow diagrams

Neil J. Calkin[a] and Marcos López de Prado[b,c,*]
[a]*Department of Mathematical Sciences, Clemson University, Clemson, SC, USA*
[b]*Senior Managing Director, Guggenheim Partners, New York, NY, USA*
[c]*Research Affiliate, Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA*

Abstract. We introduce *Stochastic Flow Diagrams* (SFDs), a new mathematical approach to represent complex dynamic systems into a single weighted digraph. This topological representation provides a way to visualize what otherwise would be a morass of equations in differences. SFDs model the *propagation* and *reverberation* that follows a shock. For example, reverberation explains how a shock to a financial system can initiate a sequence of events that lead to a crash long after the occurrence of the shock. SFDs can simulate systems in stable, steady or explosive state. SFDs add Topology to the Statistical and Econometric toolkit. We believe that SFDs will help policy makers, investors and researchers communicate and discuss better the complexity of dynamic systems.

Keywords: Time series, graph theory, topology, financial flows, macro trading

AMS Classification: 90C35, 90B15, 05C21, 62P05, 37M10

> "*Since the middle of the 20th century, theoretical physicists have increasingly turned to this tool to help them undertake critical calculations. Feynman diagrams have revolutionized nearly every aspect of theoretical physics.*"
>
> *David Kaiser (2005)*

1. Introduction

By the 1940s, theoretical Physics was stuck in the mud. The morass of calculations involved in solving simple quantum electrodynamics (QED) systems was hindering progress. Around 1948, Richard Feynman introduced a visualization technique, essentially a bookkeeping device to keep track of the structure of QED systems (Kaiser, 2005). This seemingly minor innovation, a mere sketching method, made all the difference. It allowed physicists to discuss alternative system specifications in a clear and formal new language. Feynman's idea of visualizing a complex system of equations did not solve the system itself, but it helped to talk through it in an intelligible way. Equations could always be worked out at a later stage, but the first step was to get the broad idea right.

Nobel laureate Wassily Leontief is one among several leading economists who have vocally expressed the notion that Economics has been stuck in the mud for quite some time. He diagnosed the problem in the following terms (Leontief, 1982):

> "*[E]conometricians fit algebraic functions of all possible shapes to essentially the same sets of data without being able to advance, in any perceptible way, a systematic understanding of the structure and the operations of a real economic system.*"

Old Economic disputes resurface in an apparent endless loop, and econometric models seem of little help in settling those controversies. In the words of Harvey (1997, pp. 199–200):

*Corresponding author: Marcos López de Prado, Guggenheim Partners, 330 Madison Ave., New York, NY 10017, USA. E-mail: marcos.lopezdeprado@guggenheimpartners.com; www.QuantResearch.info.

> "*[W]hat have economists learnt from fitting such models? The answer is very little. I cannot think of one article which has come up with a cointegration relationship which we did not know about already from economic theory [. . .] The solution is to combine the flexibility of a time series model with the interpretation of regression.*"

This situation presents some striking similarities with the ailments experienced by physicists around the middle of the 20th century. Part of the problem is the difficulty of discussing complex, high dimensional systems on the basis of dozens (if not hundreds) of equations simultaneously interacting with each other. It seems it would be useful to introduce a bookkeeping device *à la Feynman*, however specifically designed for representing Time Series systems. This would facilitate debates (or the interpretation of regressions, as Harvey put it), by tracking the relationships involved in competing Econometric models. In order to be successful, such device must incorporate the notion of time lapse that is essential to Time Series analysis.

Several scientific areas deal with complex systems. Computer scientists identify the possible points of failure in a network, and anticipate which routers or switches may become overloaded before a crash occurs. Operations researchers compute the path that optimally disseminates information at a minimum cost. Epidemiologists model the speed and propagation of diseases, which allows them to determine their source and establish areas of quarantine. These breakthroughs rely on recent mathematical advances in the subjects of Topology and Graph Theory.

Topology and Graph Theory are two major areas of Mathematics that emanated from the "Seven Bridges of Königsberg" problem. Around 1735, Leonhard Euler asked the question of whether it was possible to walk through the city crossing each bridge once and only once, ending at the starting point. Euler recognized that Geometry could not solve this problem, because the relevant information was not the exact geographic location of the bridges, but their connections. This focus on *connectedness* is the reason why scientists use Topology and Graph Theory to study complex systems. A practical example can be found in subway maps. These maps are not illustrations of a geographic or geometric structure, but topological representations of how lines and stations are interconnected. Should train flow be disrupted at a particular station, a passenger can use that topological map to quickly recognize what nodes must be avoided. Adding to that map geographic details would not increase our understanding of the subway system as a whole. In fact, it could be argued that adding such geographic details may obfuscate the understanding of the system. In a similar way, relying solely on linear algebra and stochastic calculus to study a system, such as the circulation of financial flows, may provide a level of detail that obfuscates some very relevant questions. For instance, linear algebra and stochastic calculus tell us very little about the number of paths and cycles in a financial network, the points of most likely congestion, what nodes could trigger a credit freeze should they shut down, etc. If statistical tests and numerical tables sufficed to understand the state of a complex system, the medical profession would not increasingly rely on CAT scans, MRI, ultrasounds and other visualization technologies to diagnose, monitor and treat patients. We are not suggesting that Statistical and Econometric applications are not useful, however we believe that Topology should be added to the Statistical and Econometric toolkit used to study systems.

The first goal of this paper is to introduce a new Graph Theory technique to visualize complex dynamic systems. This technique is general and can be applied to a wide variety of specifications and applications, not only econometric models or financial problems. Our second goal is to illustrate how this technique can help us simulate the system's response to a shock. For example, we will study how a shock can lead to a crash long after its occurrence, through the phenomenon of reverberation. Our third goal is to provide specific algorithms and computer code that carry out the calculations involved in our study.

The rest of the paper is organized as follows: Section 2 reviews the existing literature and outlines our contributions. Section 3 sets definitions and concepts useful to analyze a Time Series system. Section 4 describes how to generate topological representations of Time Series models. Section 5 discusses shock reverberation and equilibrium/disequilibrium dynamics. Section 6 presents our conclusions. The mathematical appendices explain in greater detail the arguments employed in this paper, and the Python code contains the implementation of some key aspects of our methodology.[1]

[1] All code in this paper is provided "as is", and contributed to the academic community for non-business purposes only, under a GNU-GPL license. Users explicitly renounce to any claim against the authors. The authors retain the commercial rights of any for-profit application of this software, which must be pre-authorized in written by the authors.

2. Literature and contribution

Our contribution combines elements of Graph Theory and Time Series analysis. In this section we will cite a few essential references to these two broad subjects, and explain how this paper connects them.

Graph Theory is a well-established area of Mathematical research. Graphs are one of the prime objects of study in Discrete Mathematics, and an essential tool to solve problems in Combinatorial Mathematics. There is a growing literature of advanced papers presenting Topological and Graph Theory applications to computer science and operations research. A good general treatment of this subject can be found in Bollobás (2013) or Bondy and Murty (1976). Specific treatment of random graph theory can be found in Durrett (2007). Easley and Kleinberg (2010) and Jackson (2010) discuss applications of graphs to social and economic networks. Rebonato and Denev (2014) introduce a novel Bayesian networks approach to coherent asset allocation.

Excellent references for Econometric theory and Time Series analysis would include Greene (2008), Hamilton (1994), Muirhead (1982) or Tsay (2010). Campbell, Lo and MacKinlay (1996) compile many great examples of Econometric approaches to practical financial problems.

Inspired by visualization techniques *à la Feynman*, we introduce *Stochastic Flow Diagrams* (SFDs), a new mathematical approach to represent complex systems of Time Series models in Graph form. This topological representation provides a way to visualize what otherwise would be a morass of equations in differences. Our method combines elements of Graph Theory and Inferential Statistics to visualize the structure of a complex system, allowing for an intuitive interpretation of its state and future course.

The SFD method takes into consideration the dynamic properties of the system, determining the direction of the flows in terms of lead-lag and causality effects. SFD connectivity is determined by statistical significance of the graph's arcs, which are weighted based on the estimated parameters of the Time Series model. Because SFDs map dynamic systems, they incorporate a time dimension. This is made explicit in the design of the SFD, through the definitions of *outbound arc* and *lagged vertex*.

Outbound arcs and lagged vertices are essential features of our methodology. They allow us to model memory effects. Without them, the system would instantaneously adjust to a shock, and lead-lag effects, error-corrections, trends, momentum or crashes would not be possible. Examples of systems with memory effects are economic and financial systems, where the consequences of a shock can be observed for a long period after its occurrence. This phenomenon, called *reverberation*, is a feature of our approach that is not present in studies that use contemporaneous correlation as the criterion to establish connections (see Calkin and López de Prado (2014) for a review).

In a flow system, one component may receive flow from more than one other component. For example, suppose three components a, b and c, where a receives flow from b and c. Although all relationships in a graph are by definition established in pairs, their estimation cannot be carried out in pairwise terms ($b \to a,\ c \to a$), or flow would not be additive. In a flow system, each equation determines the value of a variable as a function of multiple other variables (not only one-to-one). Accordingly, SFD connections are the solution to a full system of dynamic equations in multivariate form.

Brualdi (2010) demonstrates the mutually beneficial relationship between Graph Theory and Linear Algebra. Accordingly, we believe that SFD will help policy makers, investors and researchers communicate and discuss better the complexity of dynamic systems. For a particular application of these methods, we refer the reader to Calkin and López de Prado (2014), where we study Macro financial flows using SFDs.

3. Graph theory nomenclature

Graph Theory is a relatively new and rapidly growing subject. Its nomenclature is sometimes confusing, with different authors referring to the same concept with different names. In this section we will set out the key definitions and nomenclature used later in the paper.

3.1. Definitions

First, we begin with the definition of a graph as a set of system components (vertices), interrelated by connections (edges).

Definition 1. (Graph) A graph G is an ordered pair $G = (V_G,\ E_G)$, where V_G is a nonempty set of vertices,

and E_G is a subset of the set of unordered pairs of elements of V_G, called edges. Then, an incidence function ψ_G can be derived as the function that associates each element of E_G with an unordered pair of vertices of G.

If $e \in E_G$ and $(u,\ \upsilon) \subset V_G$, then u and υ are connected by e if and only if $\psi_G[e] = (u,\ \upsilon)$. Recall that $(u,\ \upsilon)$ is an unordered pair, which makes the graph *undirected*. This definition of a graph is not adequate for situations where the pair needs to be ordered, conveying information regarding the direction of flow. This is addressed in the next definition.

Definition 2. (Directed Graph, or Digraph) A digraph D is an ordered pair $D = (V_D,\ A_D)$, where V_D is a nonempty set of vertices, and A_D is a subset of the set of ordered $V_D x V_D$ pairs of elements, called arcs. Then, an incidence function ψ_D can be derived as the function that associates each element of A_D with an ordered pair of vertices of D.

If $a \in A_D$ and $(u,\ \upsilon) \subset V_D$, then u is connected to υ by a if and only if $\psi_D[a] = (u,\ \upsilon)$. Now $(u,\ \upsilon)$ is an ordered pair, where the arc starts in u and ends in υ. Definition 2 allows for loops (arcs that connect a vertex with itself), but multi-edges are still not allowed ($\forall u,\ \upsilon \in V_D$, where u and υ may be the same, there is at most only one $a \in A_D$ such that $\psi_D[a] = (u,\ \upsilon)$). Arcs do not need to have the same relative weight. The concept of weighted digraph introduces the possibility of channeling flows at varying rates.

Definition 3. (Weighted Digraph) A digraph D is weighted if it is endowed with a function in the real domain, $\omega_D : A_D \to \mathbb{R}$, i.e. $\forall a \in A_D,\ \exists \omega_D[a] \in \mathbb{R}$. Then, the weighted digraph is characterized by the triple $D = (V_D,\ A_D,\ \omega_D)$.

If $\forall a \in A_D,\ \omega_D[a] = k$, where k is a real constant, then the weighted digraph can be reduced to an unweighted form. It is often useful to reduce the complexity of a digraph to a portion of it, a concept formally defined as subdigraph.

Definition 4. (Subdigraph) D' is a subdigraph of D if $V_{D'} \subseteq V_D$, $A_{D'} \subseteq A_D$ and the elements of $A_{D'}$ are edges relative to the vertex set D'. Then, $\psi_{D'}$ is the restriction of ψ_D to $A_{D'}$.

Given a digraph, we can use alternative definitions of routes crossing through its vertices and arcs.

Definition 5. (Walk, Trail, Path, Cycle) A walk in D between two vertices $u_0,\ u_k$ is a finite non-null sequence $W[u_0,\ u_k] = u_0,\ a_1,\ u_1,\ a_2,\ \ldots,\ u_k$, whose terms are alternately vertices and arcs, such that, for $1 \leq i \leq k$, the ends of a_i are u_{i-1} and u_i. The length of the walk is the integer k. A trail is a walk where the arcs $a_1,\ \ldots,\ a_k$ are distinct. A path is a trail where the vertices $u_0, \ldots, u_k$ are also distinct. A cycle is a path where $u_0 = u_k$.

A path with no repeated vertices is called a *simple path*, and a cycle with no repeated vertices or edges (aside from the necessary repetition of the start and end vertex) is a *simple cycle*. The vertices of a cycle are said to be *strongly connected*. A path that visits every arc exactly once is called an *Eulerian path*. An *Eulerian cycle* is an Eulerian path that starts and ends on the same vertex. A path that visits every vertex exactly once is called a *Hamiltonian path*. A *Hamiltonian cycle* is a Hamiltonian path that is a cycle. Thanks to the above concepts, we can establish a measure of distance.

Definition 6. (Distance, Diameter) Given $a \in A_D$ and $(u,\ \upsilon) \subset V_D$, the distance between vertices u and υ, $d_D[u,\ \upsilon]$, is the sum of the weights across a shortest (not necessarily unique) path between them. $d_D[u,\ u] = 0$. When u and υ are unreachable from each other, $d_D[u,\ \upsilon] = \infty$. The diameter of V_D is the maximum distance between any two vertices in it, $\max_{u,\ \upsilon \in D} d_D[u,\ \upsilon]$.

Definition 7. (Density, Degree of Centrality) The density of a digraph D is defined as $d = \frac{m}{n(n-1)}$, where n is the number of vertices and m is the number of arcs. The degree of centrality of a vertex is the fraction of nodes it is connected to.

The notion behind this definition of density is to compute the average number of arcs per possible pair of vertices.

Definition 8. (Neighborhood, Clustering) The neighborhood of a vertex $\upsilon_i \in V_D$ is defined as $N_i = \left\{\upsilon_j \in V_D \,\middle|\, (\upsilon_i,\ \upsilon_j) \in A_D \wedge (\upsilon_j,\ \upsilon_i) \in A_D\right\}$. Its clustering coefficient is defined as $C_i = \frac{|\{(\upsilon_j,\ \upsilon_k) | \upsilon_j,\ \upsilon_k \in N_i \wedge (\upsilon_j,\ \upsilon_k) \in A_D\}|}{|N_i|(|N_i|-1)}$, and $|.|$ is the function that counts the number of items in a set.

Definition 9. (Adjacency, Local Vertex Connectivity) Given to vertices $(u,\ \upsilon) \subset V_D$, they are adjacent if and only if $\exists a \in A_D \,|\psi_D\,[a] = (u,\ \upsilon)$. The local vertex

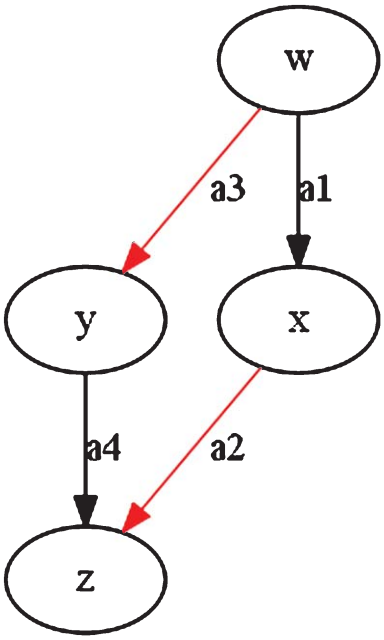

Fig. 1. Example of a directed graph.

connectivity of two non-adjacent vertices, $k_D[u,\ \upsilon]$, is defined as the number of vertices that must be removed (along with their incident arcs) to disconnect u and υ. If u and v are adjacent, then their local vertex connectivity is defined as $k_D[u,\ \upsilon] = k_{D'}[u,\ \upsilon] + 1$, where D' is the subdigraph of D that excludes the arcs connecting u and v.

Intuitively, the clustering coefficient gives us the proportion of arcs within its neighborhood divided by the number of possible arcs that could exist between them. This is a measure of the degree to which vertices tend to group together.

3.2. An example

Before we move forward, it may be helpful to see how these concepts can be applied to describe a Time Series system. Suppose a financial system with four variables: w, x, y and z. All variables are positively related. Flow can propagate following the path $w \to x \to z$ or $w \to y \to z$. In terms of graph theory, this is represented as $D = (V_D,\ A_D)$, where $V_D\{w,\ x,\ y,\ z\}$ and $A_D = \{(w,\ x),\ (x,\ z),\ (w,\ y),\ (y,\ z)\}$. Labeling the elements of $A_D = \{a_1,\ a_2,\ a_3,\ a_4\}$, the incidence function can be derived as $\psi_D[a_1] = (w,\ x)$, $\psi_D[a_2] = (x, z)$, $\psi_D[a_3] = (w, y)$, $\psi_D[a_4] = (y, z)$. Furthermore, we could assign some weights to the arcs, whereby $\omega_D[a_1] = 1$, $\omega_D[a_2] = -1$, $\omega_D[a_3] = -1$, $\omega_D[a_4] = 1$. Figure 1 pictures the associated weighted digraph. Consider the alternative digraph $D' = (V_{D'},\ A_{D'})$, where $A_{D'} = \{a_1,\ a_2\}$. Then, D' is a subdigraph of D, because D' is contained in D, and $\psi_{D'}$ is a restriction of ψ_D to $A_{D'}$. The walk between w and z is given by $W[w,\ z] = w,\ a_1,\ x,\ a_2,\ z$, which has length 2. Because arcs a_1, a_2 are distinct, $W[w,\ z]$ is also a trail. Because w and z are distinct, $W[w,\ z]$ is also a path. $W[w,\ z]$ is an Eulerian path of D', but not of D. Walk $W[w,\ z]$ has a distance of 0. Given the arcs' direction, the digraph has infinite diameter, and there are no cycles.

We make extensive use of these definitions in Calkin and López de Prado (2014), and refer the reader to that publication for further examples.

4. Topological representation of time series models

In this section we will demonstrate how to construct Stochastic Flow Diagrams (SFDs), which are complex systems of Time Series models topologically represented as weighted digraphs. We will concentrate on the type of topology needed for most applications, however the concept of SFDs can be extended to other types of Time Series.

4.1. Autoregressive processes

Let $\{y_t\}$ be a series of real-valued random variables, indexed by time t. An autoregressive model (AR) is a mathematical model of a random variable that is self-excited through a feedback mechanism. In its simplest form, it is characterized by a one-lag equation (or AR(1)),

$$y_t = c + \varphi_1 y_{t-1} + \varepsilon_t \tag{1}$$

where φ_1 is the real-valued factor that scales the lagged observation, c is a real-valued constant, and ε_t is a random variable distributed as white noise. Without loss of generality, we shall assume that $c = 0$ after centering the variables. We would like to represent the information contained in Eq. (1) into a graph. We create a weighted digraph $D = (V_D,\ A_D,\ \omega_D)$, where

- $V_D = \{\upsilon_0,\ \upsilon_1\}$
- $A_D = \{o_0,\ a_1\}$
- $\psi_D[o_0] = (\upsilon_0,\ \upsilon_1)$
  $\psi_D[a_1] = (\upsilon_1,\ \upsilon_0)$
- $\omega_D[o_0] = 1$
  $\omega_D[a_1] = \varphi_1$

The key concept to grasp is that vertices are associated with values (or states) of random variables. For example, vertex υ_0 is associated with value y_t, and vertex υ_1 is associated with value y_{t-1}. The time structure is incorporated through the concept of *lagged vertex*.

Definition 10. (Lagged and Current vertices) Consider a vertex $\upsilon_k \in V_D$ that is associated with a variable lagged k times, y_{t-k}, where $k \geq 0$. If $k > 0$, υ_k is a lagged vertex, and otherwise it is a current vertex.

There is one vertex for each variable, lagged or current. We will represent current variables with elliptical vertices, and lagged variables with diamond-shaped vertices. Vertices are connected by arcs, which can be outbound or inbound.

Definition 11. (Outbound arc) Given two vertices $\upsilon_k,\ \upsilon_{k+1} \in V_D$, an arc $o_k \in A_D$ is outbound if it connects a vertex υ_k to a vertex υ_{k+1}, with $\omega_D[o_k] = y_{t-k}$.

Definition 12. (Inbound arc) Consider two vertices $\upsilon_k,\ u_0 \in V_D$, such that vertex υ_k is associated with variable y_{t-k}, and vertex u_0 is associated with variable x_t. Provided that x_t and y_t are different variables, an arc $a_k \in A_D$ is inbound if it connects a vertex υ_k to a vertex u_0, with $\omega_D[a_k] = \varphi_k y_{t-k}$ and $k \geq 0$. If x_t and y_t are the same variable, then the condition is that $k > 0$, to avoid self-loops.

Arc o_0 is *outbound* because it takes flow away from υ_0 towards *lagged vertex* υ_1 at a weight 1, thus adding a lag. Outbound arcs always have unit weight, and there are as many as the number of lags required by the model. We will represent this type of arc with a dashed arrow. In contrast, arc a_1 is *inbound* because it goes from υ_1 to υ_0, returning the flow to the current variable. There are as many inbound arcs as regression coefficients, which determine their weight, like $\omega_D[a_1] = \varphi_1$ in the case above. We will represent inbound arcs with a solid arrow. In other words, digraph D is replicating the lag structure in Eq. (1) by using vertex υ_1 as a memory cell that receives (through *outbound* arc o_0) and returns (through *inbound* arc a_1) flow as needed. Different colors can be used to represent various arc weights and vertex values. Figure 2 represents AR(1) as a weighted digraph, D.

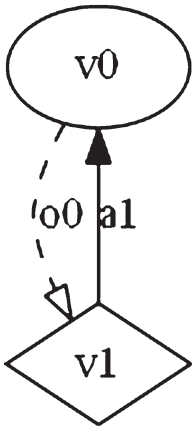

Fig. 2. SFD representation of an AR(1) process.
Note: Each variable is associated with one vertex. An ellipse-shaped vertex signals a *current variable*, and a diamond-shaped vertex signals a *lagged variable*. An *outbound arc* connects a current with a lagged variable with unit weight, and it is represented with a dashed arrow. An inbound arc connects a (current or lagged) variable with a current variable, and it is represented with a solid arrow. Its weight is given by the regression coefficient that links both variables in the AR(1) equation. The user can report different arc weights and vertex values with alternative colors.

Lags beyond one can have an effect on y. Let's consider now the case with two lags, as expressed in the AR(2) equation

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \varepsilon_t \tag{2}$$

which leads to a weighted digraph $D = (V_D,\ A_D,\ \omega_D)$, where

- $V_D = \{\upsilon_0,\ \upsilon_1,\ \upsilon_2\}$
- $A_D = \{o_0,\ o_1,\ a_1,\ a_2\}$
- $\psi_D[o_0] = (\upsilon_0,\ \upsilon_1)$
  $\psi_D[o_1] = (\upsilon_1,\ \upsilon_2)$
  $\psi_D[a_1] = (\upsilon_1,\ \upsilon_0)$
  $\psi_D[a_2] = (\upsilon_2,\ \upsilon_0)$ (3)
- $\omega_D[o_0] = 1$
  $\omega_D[o_1] = 1$

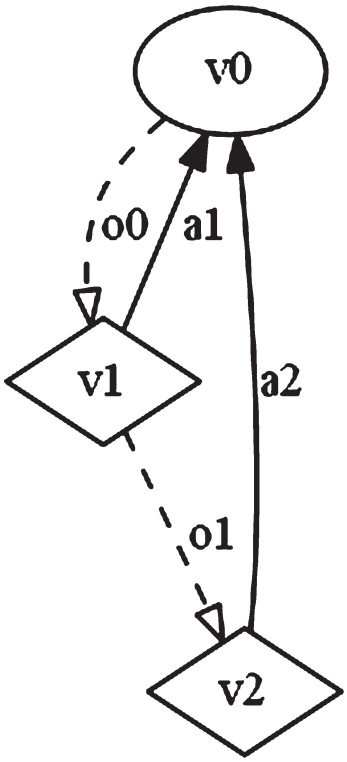

Fig. 3. SFD representation of an AR(2) process.
Note: Since the process involves a variable lagged twice, there is one elliptical vertex (current variable) and two diamond vertices (lagged variables). Two outbound (dashed) arcs drive flow towards lagged variables, and two inbound (solid) arcs drive it back towards the elliptical vertex.

$$\omega_D[a_1] = \varphi_1$$
$$\omega_D[a_2] = \varphi_2$$

Figure 3 illustrates the representation of AR(2) as a weighted digraph, D.

In general terms, an AR(p) process is characterized by the p-lag equation

$$y_t = \sum_{k=1}^{p} \varphi_k y_{t-k} + \varepsilon_t \tag{4}$$

where p is the integer number of lags, φ_k is the real-valued factor that scales the kth lagged observation, and ε_t is a random variable distributed as white noise. We would like to represent the information contained in Eq. (4) into a graph. We construct a digraph $D = (V_D,\ A_D,\ \omega_D)$, where for $k = 0,\ \ldots,\ p-1$:

- $V_D = \{\upsilon_k\}$
- $A_D = \{a_{k+1}\} \cup \{o_k\}$
- $\psi_D[a_{k+1}] = (\upsilon_{k+1},\ \upsilon_0)$
  $\psi_D[o_k] = (\upsilon_k,\ \upsilon_{k+1})$ (5)
- $\omega_D[a_{k+1}] = \varphi_{k+1}$
  $\omega_D[o_k] = 1$

Figure 4 draws the SFD associated with AR(3), AR(4), AR(5) and AR(10) processes. Appendix 2 contains the implementation of Eq. (5) in Python language.

4.2. Vector autoregressive systems

The concept of AR models is extended to a system of equations in Vector AutoRegression (VAR) models. Consider a VAR(1) model on 2 variables

$$y_{1,t} = \varphi_{1,1,1} y_{1,t-1} + \varphi_{1,2,1} y_{2,t-1} + \varepsilon_{1,t}$$
$$y_{2,t} = \varphi_{2,1,1} y_{1,t-1} + \varphi_{2,2,1} y_{2,t-1} + \varepsilon_{2,t} \tag{6}$$

where $\varphi_{i,j,k}$ is the regression coefficient associated with equation i, regressor j, after k lags. The error terms satisfy the conditions of a generalized white noise (in short, $E[\varepsilon_{i,t}] = 0$; $E[\varepsilon_{i,t}\varepsilon_{j,t}] = \Omega_{i,j}$ such that the nxn matrix formed by $\{\Omega_{i,j}\}$ is symmetric positive definite; $E[\varepsilon_{i,t}\varepsilon_{j,t-\delta}] = 0, \forall i,\ j = \{1, \ldots,\ n\}$ and any non-zero integer δ. See Hamilton (1994, pp. 257–258) for details). Following the procedure introduced earlier, the topological representation of Eq. (6) consist in constructing a weighted digraph $D = (V_D,\ A_D,\ \omega_D)$, where:

- $V_D = \{\upsilon_{1,0},\ \upsilon_{1,1},\ \upsilon_{2,0},\ \upsilon_{2,1}\}$
- $A_D = \{o_{1,0},\ o_{2,0},\ a_{1,1,1},\ a_{1,1,2},\ a_{2,1,1},\ a_{2,1,2}\}$
- $\psi_D[o_{1,0}] = (\upsilon_{1,0},\ \upsilon_{1,1})$
  $\psi_D[o_{2,0}] = (\upsilon_{2,0},\ \upsilon_{2,1})$
  $\psi_D[a_{1,1,1}] = (\upsilon_{1,1},\ \upsilon_{1,0})$
  $\psi_D[a_{1,1,2}] = (\upsilon_{1,1},\ \upsilon_{2,0})$
  $\psi_D[a_{2,1,1}] = (\upsilon_{2,1},\ \upsilon_{1,0})$
  $\psi_D[a_{2,1,2}] = (\upsilon_{2,1},\ \upsilon_{2,0})$ (7)
- $\omega_D[o_{1,0}] = 1$
  $\omega_D[o_{2,0}] = 1$
  $\omega_D[a_{1,1,1}] = \varphi_{1,1,1}$

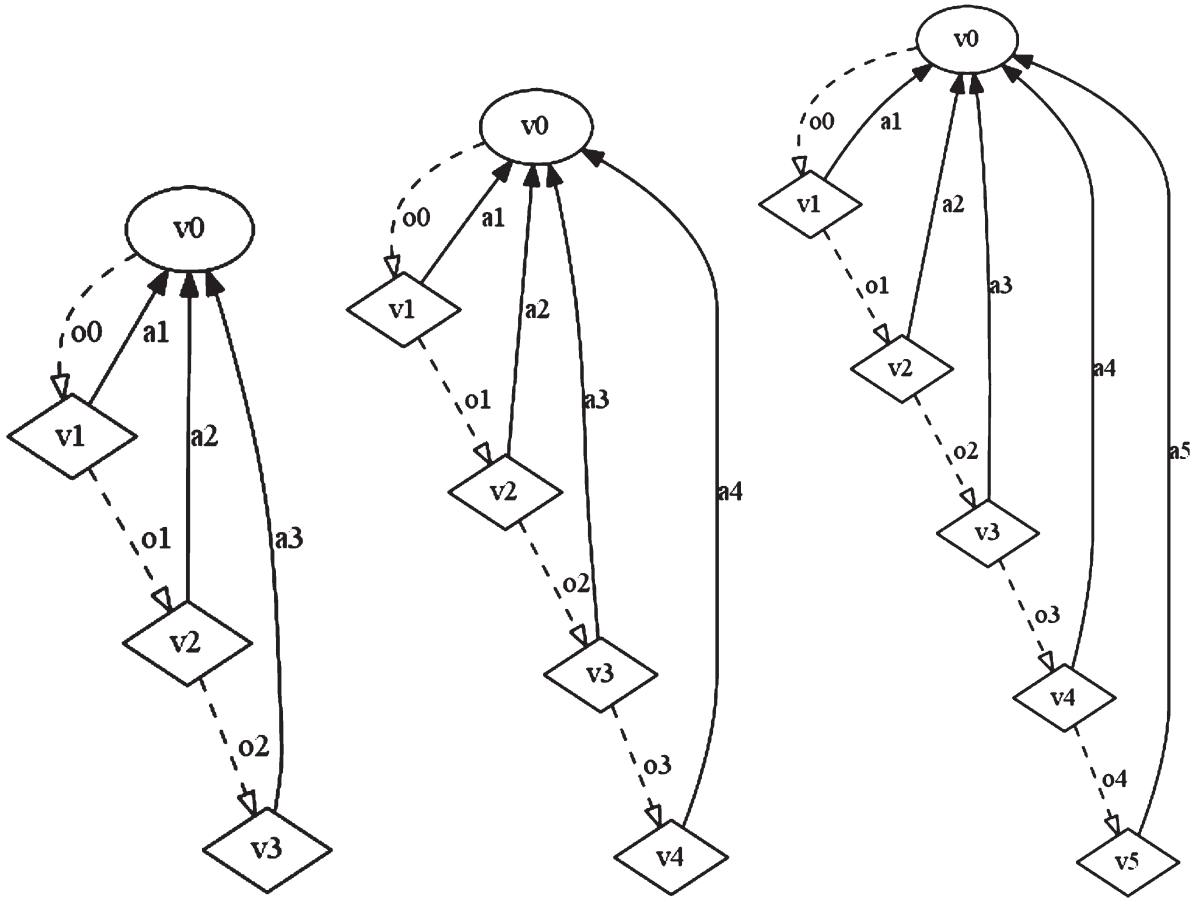

Fig. 4.1–4.3. From left to right, SFD representation of AR(3), AR(4), AR(5).
Note: Because AR(p) are single equation processes, they have a single elliptical vertex (current variable), with as many diamond vertices as lagged variables. Each inbound arc (solid arrow) is associated with one equation parameter (which determines its weight), and each equation parameter is associated with a single inbound arc. There are as many outbound arcs as diamond vertices, all with unit weight.

$$\omega_D[a_{1,1,2}] = \varphi_{2,1,1}$$

$$\omega_D[a_{2,1,1}] = \varphi_{1,2,1}$$

$$\omega_D[a_{2,1,2}] = \varphi_{2,2,1}$$

In Section 4.1, we dealt with only one equation, which allowed us to simplify the notation. Working with 2 equations has required the adoption of a more compact notation. In Eq. (7) we denote as $\upsilon_{i,k}$ the vertex associated with variable i and lag k ($y_{i,t-k}$). Arc $o_{i,k}$ denotes the connection that begins in vertex $\upsilon_{i,k}$ and ends in vertex $\upsilon_{i,k+1}$ (outbound arc), thus lagging the variable. Once the flow has been sufficiently lagged, an inbound arc $a_{i,k,j}$ channels it back, from vertex $\upsilon_{i,k}$ to vertex $\upsilon_{j,0}$, where $k > 0$ and i and j may be equal or different (inbound arc). The topological representation of this system is charted in Fig. 5.

Consider now a VAR(p) system on two equations, like the one in Eq. (8).

$$\begin{aligned} y_{1,t} &= \varepsilon_{1,t} + \sum_{k=1}^{p} \varphi_{1,1,k} y_{1,t-k} + \varphi_{1,2,k} y_{2,t-k} \\ y_{2,t} &= \varepsilon_{2,t} + \sum_{k=1}^{p} \varphi_{2,1,k} y_{1,t-k} + \varphi_{2,2,k} y_{2,t-k} \end{aligned} \quad (8)$$

This system can be represented as a weighted digraph $D = (V_D, A_D, \omega_D)$, whereas for $k = 0, \ldots, p-1$ we set:

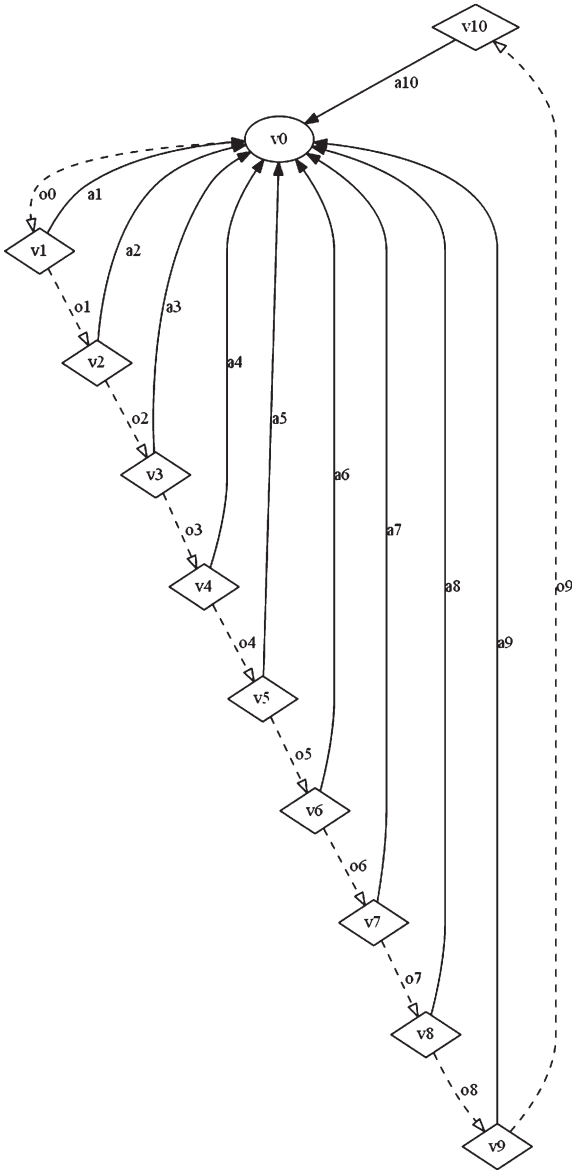

Fig. 4.4. SFD representation of a AR(10) process.

- $V_D = \{\upsilon_{1,k},\ \upsilon_{2,k}\}$
- $A_D = \{a_{1,k+1,1}\} \cup \{a_{2,k+1,1}\} \cup \{a_{1,k+1,2}\}$
  $\cup \{a_{2,k+1,2}\} \cup \{o_{1,k},\ o_{2,k}\}$
- $\psi_D[a_{1,k+1,1}] = (\upsilon_{1,k+1},\ \upsilon_{1,0})$
  $\psi_D[a_{2,k+1,1}] = (\upsilon_{2,k+1},\ \upsilon_{1,0})$
  $\psi_D[a_{1,k+1,2}] = (\upsilon_{1,k+1},\ \upsilon_{2,0})$
  $\psi_D[a_{2,k+1,2}] = (\upsilon_{2,k+1},\ \upsilon_{2,0})$
  $\psi_D[o_{1,k}] = (\upsilon_{1,k},\ \upsilon_{1,k+1})$
  $\psi_D[o_{2,k}] = (\upsilon_{2,k},\ \upsilon_{2,k+1})$ (9)
- $\omega_D[a_{1,k+1,1}] = \varphi_{1,1,k+1}$
  $\omega_D[a_{2,k+1,1}] = \varphi_{1,2,k+1}$
  $\omega_D[a_{1,k+1,2}] = \varphi_{2,1,k+1}$
  $\omega_D[a_{2,k+1,2}] = \varphi_{2,2,k+1}$
  $\omega_D[o_{1,k}] = 1$
  $\omega_D[o_{2,k}] = 1$

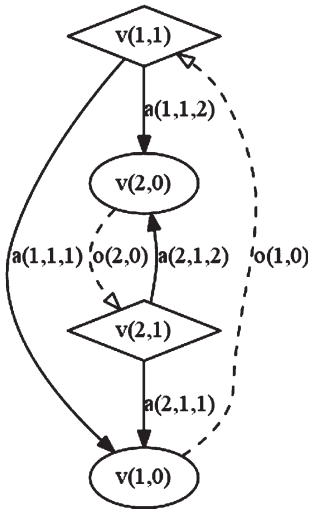

Fig. 5. SFD representation of a VAR(1) system on two variables.
Note: Without looking at the equation behind this diagram, it is obvious that it is a VAR system. AR processes cannot involve more than one elliptical vertex. We know that this is a VAR(1) system on two equations because there are two elliptical vertices, and only one diamond-shaped vertex lagging each contemporaneous variable.

Figure 6 draws the SFD associated with VAR(2), VAR(3), VAR(5) and VAR(10) systems on two variables. As it can be appreciated, merely adding a lag to a VAR model substantially increases its complexity. Also, the architecture of SFD representations of VAR(p) models is much more complex and less hierarchical than the SFD representations of their AR(p) counterparts.

Finally, we will show how to translate into SFDs a VAR(p) model on any number n of equations, characterized by Eq. (10).

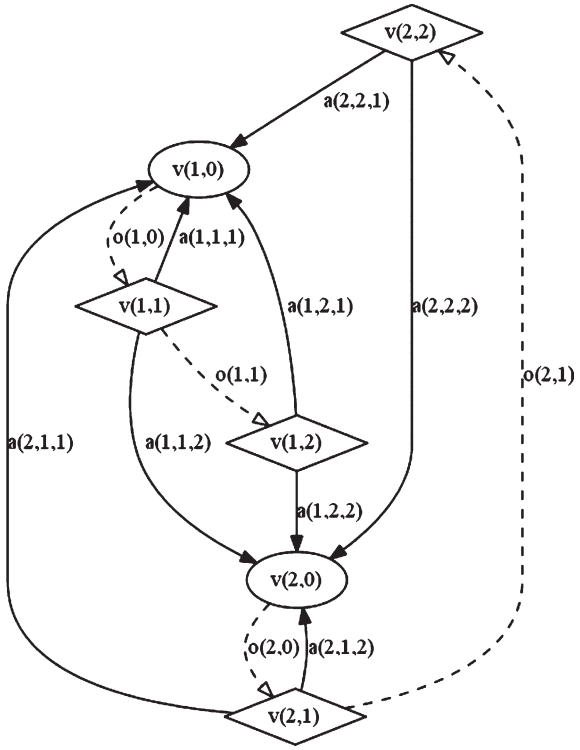

Fig. 6.1. SFD representation of a VAR(2) system on two variables. *Note:* As this diagram illustrates, simply adding one lag to a VAR system substantially increases the complexity of the model, much more so than it was in the case of AR(p) specifications.

$$\begin{aligned} y_{1,t} &= \sum_{j=1}^{n}\sum_{k=1}^{p}\varphi_{1,j,k}y_{j,t-k} + \varepsilon_{1,t} \\ &\cdots \\ y_{n,t} &= \sum_{j=1}^{n}\sum_{k=1}^{p}\varphi_{n,j,k}y_{j,t-k} + \varepsilon_{n,t} \end{aligned} \tag{10}$$

The system can be represented as a weighted digraph $D = (V_D,\ A_D,\ \omega_D)$. For $i,\ j = 1, \ldots, n$ and $k = 0, \ldots, p-1$ we set:

- $V_D = \{\upsilon_{i,k}\}$
- $A_D = \{a_{i,k+1,j}\} \cup \{o_{i,k}\}$
- $\psi_D\,[a_{i,k+1,j}] = (\upsilon_{i,k+1},\ \upsilon_{j,0})$
  $\psi_D\,[o_{i,k}] = (\upsilon_{i,k},\ \upsilon_{i,k+1})$ (11)
- $\omega_D[a_{i,k+1,j}] = \varphi_{j,i,k+1}$
  $\omega_D[o_{i,k}] = 1$

Figure 7 draws the SFD associated with VAR(1) and VAR(2) system on three variables. Figure 8 draws the SFD associated with a VAR(1) system on five variables. Appendix 3 contains the implementation of Eq. (11) in Python language. Because the left-hand side variables in each equation are not explanatory variables on any other equation, the system is not simultaneously determined, thus each equation can be fitted individually by ordinary least squares (OLS) (see Zellner (1962), Greene (2008, p. 696)).

4.3. Structural vector autoregression systems

Consider now a Structural Vector AutoRegression (SVAR) model. This is a system that combines synchronous variables with lagged variables, such as the one characterized by Equation (12):

$$\begin{aligned} y_{1,t} &= \sum_{j=2}^{n}\varphi_{1,j,0}y_{j,t} + \sum_{j=1}^{n}\sum_{k=1}^{p}\varphi_{1,j,k}y_{j,t-k} + \varepsilon_{1,t} \\ &\cdots \\ y_{i,t} &= \sum_{\substack{j=1 \\ j\neq i}}^{n}\varphi_{i,j,0}y_{j,t} + \sum_{j=1}^{n}\sum_{k=1}^{p}\varphi_{i,j,k}y_{j,t-k} + \varepsilon_{i,t} \\ &\cdots \\ y_{n,t} &= \sum_{j=1}^{n-1}\varphi_{n,j,0}y_{j,t} + \sum_{j=1}^{n}\sum_{k=1}^{p}\varphi_{n,j,k}y_{j,t-k} + \varepsilon_{n,t} \end{aligned} \tag{12}$$

The system can be represented as a weighted digraph $D = (V_D,\ A_D,\ \omega_D)$. For $i,\ j = 1, \ldots, n$ and $k = 0, \ldots, p-1$ we set:

- $V_D = \{\upsilon_{i,k}\}$
- $A_D = \{a_{i,0,j} | i \neq j\} \cup \{a_{i,k+1,j}\} \cup \{o_{i,k}\}$
- $\psi_D[a_{i,0,j}] = (\upsilon_{i,0},\ \upsilon_{j,0})$, where $i \neq j$
  $\psi_D[a_{i,k+1,j}] = (\upsilon_{i,k+1},\ \upsilon_{j,0})$
  $\psi_D[o_{i,k}] = (\upsilon_{i,k},\ \upsilon_{i,k,+1})$ (13)
- $\omega_D[a_{i,0,j}] = \varphi_{j,i,0}$, where $i \neq j$
  $\omega_D[a_{i,k+1,j}] = \varphi_{j,i,k+1}$
  $\omega_D[o_{i,k}] = 1$

Figure 9 displays the SFD associated with SVAR systems for $(p, n) = (1, 2)$ and $(p, n) = (2, 2)$. Inbound arcs connecting two elliptical vertices sig-

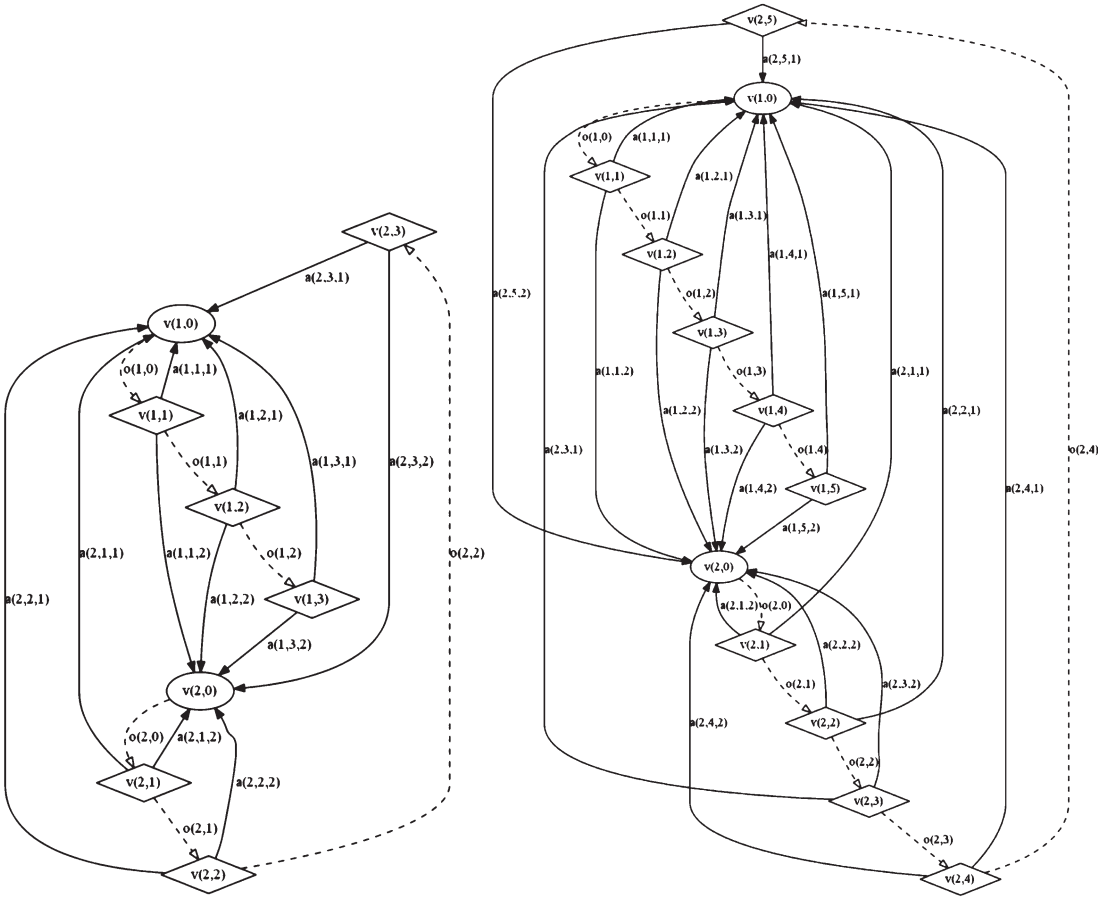

Fig. 6.2, 6.3. From left to right, SFD representation of VAR(3) and VAR(5) systems on two variables.
Note: Regardless of how complex a diagram may look, it is easy to recognize the system of equations behind it. The number of equations is given by the number of elliptical vertices, and looking at a particular variable, counting the number of diamond-shaped vertices tells us the lags involved. Here we represent all inbound arcs, but in practice these diagrams will be simpler as a result of statistically insignificant regression coefficients, thus highlighting what connections are truly relevant and which are not.

nal the presence of contemporaneous effects. To make these contemporary effects more obvious, we draw the involved inbound arcs with a flat arrowhead. Appendix 4 contains the implementation of Eq. (13) in Python language.

Appendix 5 summarizes the SFD symbology employed so far. Our procedure is not limited to AR, VAR or SVAR specifications. Many other dynamic specifications are also amenable to this approach. For example, Error Correction systems will have vertices associated with the difference between pairs of lagged variables. ARMA systems will have vertices holding the moving average components, with the relevant connections just as we did in this section with autoregressive components.

5. Shock propagation and reverberation

Each of the SFD's vertices represents a random variable. Their random nature derives from the unpredictability of the shocks that disturb them. *Propagation* is the transmission of the shock through the network, via inbound arcs. Because lagged vertices introduce memory effects, shocks are not instantaneously resolved. *Reverberation* is the persistence of the shock after its occurrence. A shock reverberates over time, generating waves of flows that further disturb the system, which result in complex interactions. Depending on the configuration of the system, reverberation can fade over time (stable state), persist indefinitely (steady) or generate a crash (explosive state). In this

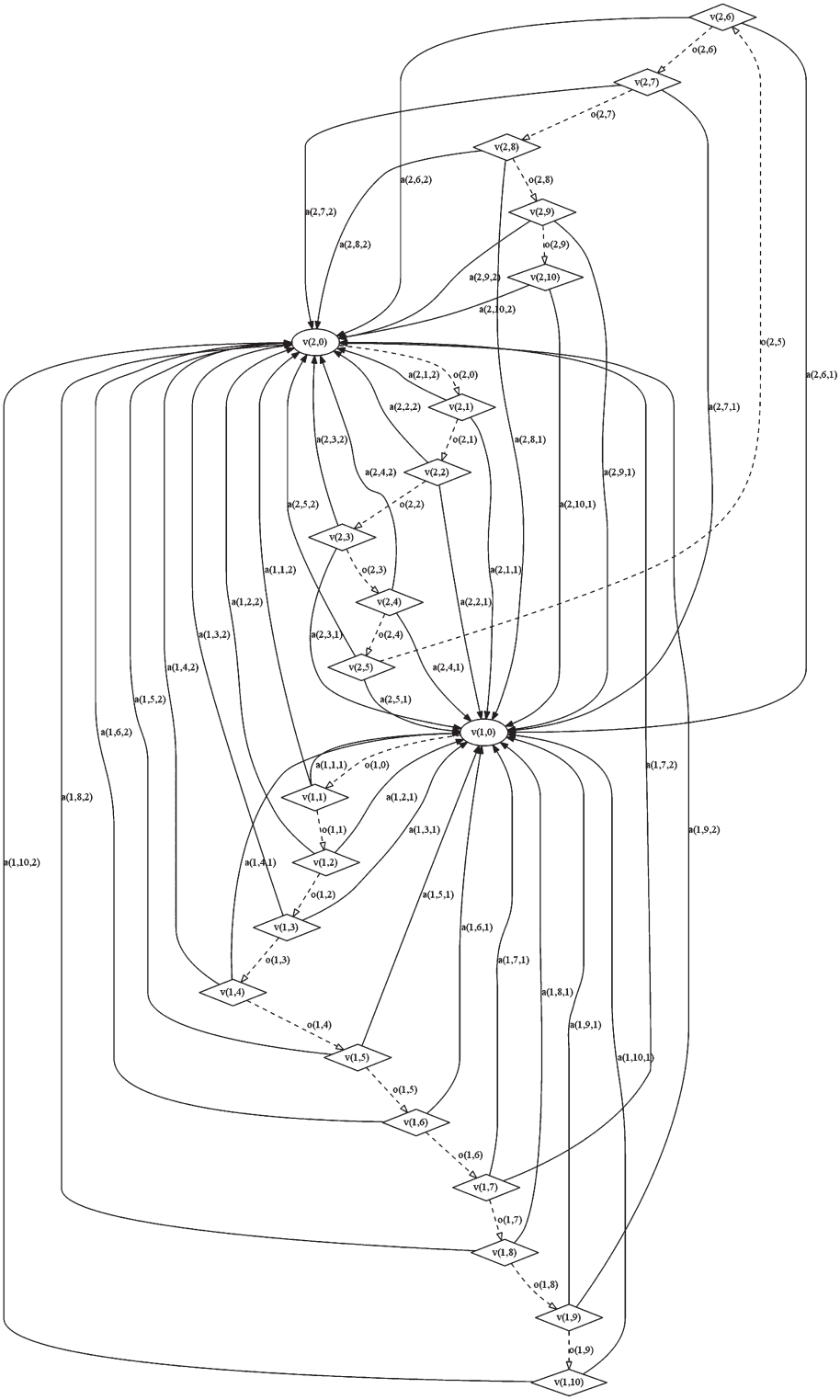

Fig. 6.4. SFD representation of a VAR(10) system on two variables.

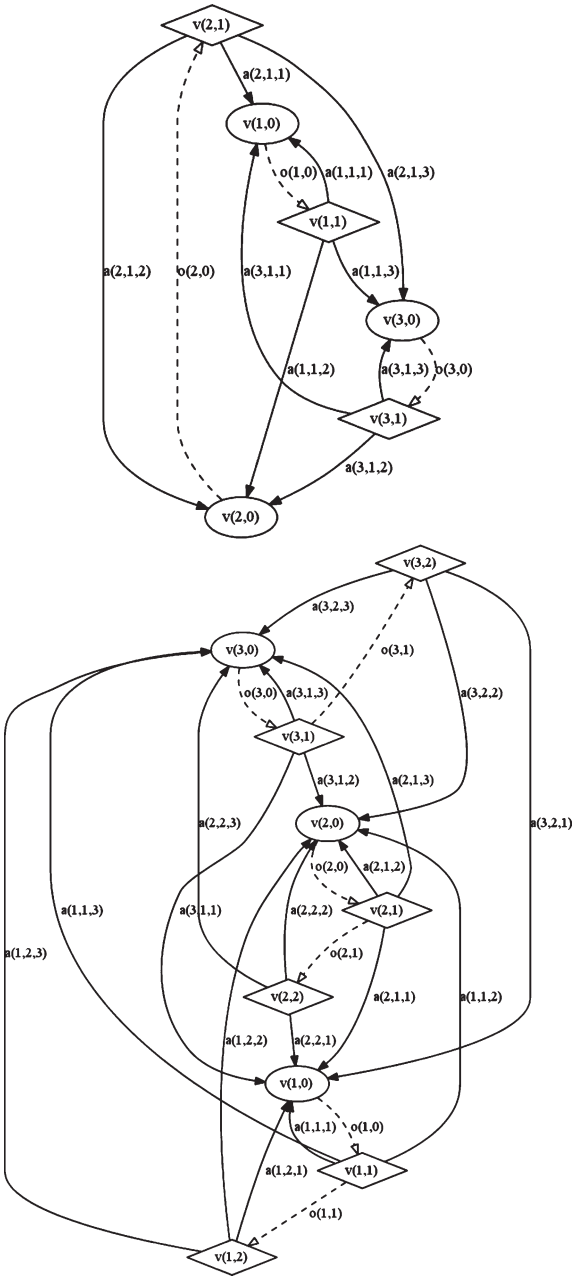

Fig. 7.1, 7.2. From left to right, SFD representation of VAR(1) and VAR(2) systems on three variables.

section we will illustrate how SFDs can help visualize the outcomes of these shocks.

A VAR(1) system on two equations provides us with the simplest case of reverberation. In Appendix 6 we have derived the conditions under which such system can be *stable*, *steady* or *explosive*. Solving the equivalent equations for large systems is not always practical, in which case SFDs constitute a valuable tool to study the response of a system to specific shocks.

5.1. Stable state

Consider a VAR(1) system on two equations, where each variable is positively correlated with its own lagged value, and negatively correlated with the lagged value of the other variable. One example would be a financial system where stocks and bonds have momentum, and each has a reverse effect on the other (if stocks go up today, bonds will tend to go down tomorrow, and vice versa). This situation can be characterized with the following parameters matrix:

$$\Phi = \begin{bmatrix} \varphi_{1,1,1} & \varphi_{1,2,1} \\ \varphi_{2,1,1} & \varphi_{2,2,1} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{bmatrix} \tag{14}$$

Following our analysis in Appendix 6, this parameters matrix has eigenvalues $\Lambda_{1,1} = \frac{tr[\Phi]+\sqrt{(tr[\Phi])^2-4|\Phi|}}{2} = \frac{1}{2}$ and $\Lambda_{2,2} = \frac{tr[\Phi]-\sqrt{(tr[\Phi])^2-4|\Phi|}}{2} = 0$.

The conclusion is that, no matter how strong shocks are, this system should always stabilize and slowly converge to a new equilibrium. We can use SFDs to simulate shocks and observe how the system evolves over time. For instance, suppose that 'Stocks' experience a negative shock, $\varepsilon_{1,0} = -\frac{1}{2}$. Figure 10 draws the corresponding SFDs after 1, 5 and 10 periods, following the same symbology discussed in Appendix 5.

In a stable state, the system absorbs the shock and new equilibrium levels are reached after a few periods. The rate of change fades over time, as indicated by the color of the vertices' left half. The cumulative effect converges to the new equilibrium level, as evidenced by the color of the vertices' right half. Note that stocks are much more affected than bonds.

5.2. Steady state

Consider a VAR(1) system on two equations, with the following parameters:

$$\Phi = \begin{bmatrix} \varphi_{1,1,1} & \varphi_{1,2,1} \\ \varphi_{2,1,1} & \varphi_{2,2,1} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{15}$$

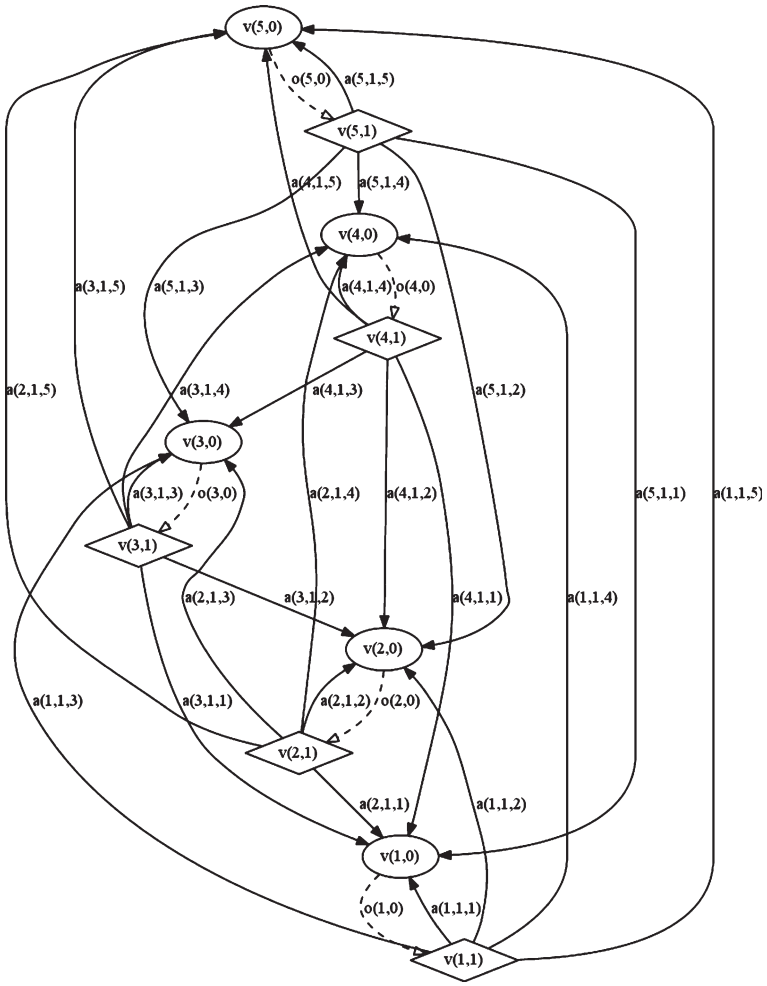

Fig. 8. SFD representation of a VAR(1) system on five variables.

This parameters matrix has eigenvalues $\Lambda_{1,1} = \frac{tr[\Phi]+\sqrt{(tr[\Phi])^2-4|\Phi|}}{2} = 1$ and $\Lambda_{2,2} = \frac{tr[\Phi]-\sqrt{(tr[\Phi])^2-4|\Phi|}}{2} = 0$.

Like in the case earlier, suppose that 'Stocks' experience a negative shock, $\varepsilon_{1,0} = -\frac{1}{2}$. Figure 11 draws the corresponding SFDs after 1, 5 and 10 periods.

In a steady state, the system is unable to absorb the shock, and a drift continues until another shock counters it. The rate of change does not fade over time, as indicated by the color of the vertices' left half. The cumulative effect grows indefinitely, as evidenced by the color of the vertices' right half. Because the drift continues, bonds also end up being significantly affected.

5.3. Explosive state

Consider a VAR(1) system on two equations, with the following parameters:

$$\Phi = \begin{bmatrix} \varphi_{1,1,1} & \varphi_{1,2,1} \\ \varphi_{2,1,1} & \varphi_{2,2,1} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} \end{bmatrix} \tag{16}$$

This parameters matrix has eigenvalues $\Lambda_{1,1} = \frac{tr[\Phi]+\sqrt{(tr[\Phi])^2-4|\Phi|}}{2} = \frac{4}{3}$ and $\Lambda_{2,2} = \frac{tr[\Phi]-\sqrt{(tr[\Phi])^2-4|\Phi|}}{2} = 0$.

Following earlier examples, suppose that 'Stocks' experience a negative shock, $\varepsilon_{1,0} = -\frac{1}{2}$. Figure 12 draws the corresponding SFDs after 1, 5 and 10

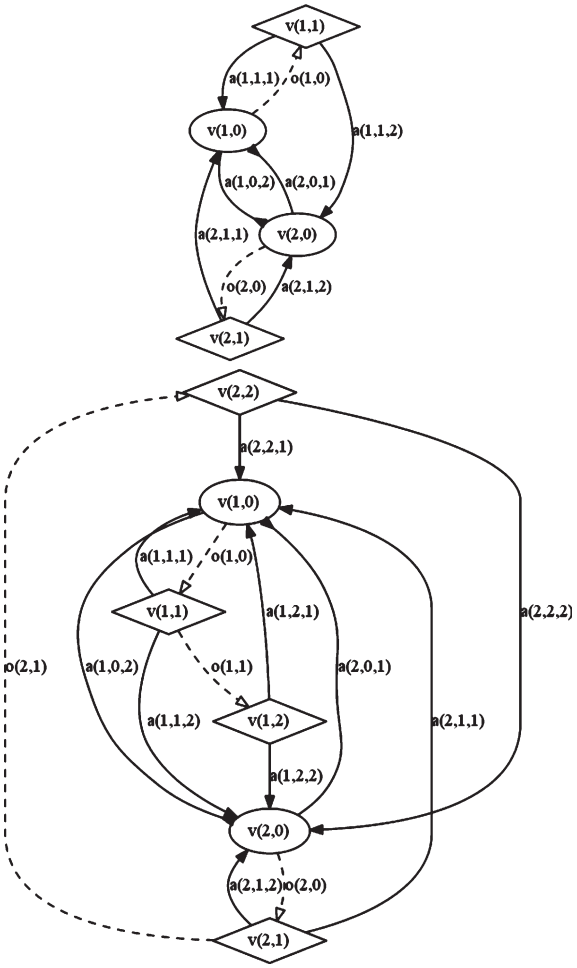

Fig. 9. SFDs associated with SVAR systems for $(p,\ n) = (1,\ 2)$ and $(p,\ n) = (2,\ 2)$.
Note: It is evident from these diagrams that the systems of equations behind them cannot be VAR. The reason is, there are inbound arcs connecting two elliptical vertices, signaling the presence of contemporaneous effects. To make these more obvious, those inbound arcs are drawn with a flat arrowhead. Hence, these diagrams represent SVAR systems.

periods. In an explosive state, the system amplifies the shock over time. The rate of change increases after each period, as indicated by the color of the vertices' left half. The cumulative value explodes after a few periods, as evidenced by the color of the vertices' right half. The entire system is disrupted and eventually crashes.

SFDs make it easy to simulate alternative scenarios, like studying the effects of combined shocks, or shock propagation after manually adding or removing one of the arcs, based on the researcher's fundamental believe that such arc will play a different role going forward.

6. Conclusions

Visualization techniques have proven extremely valuable in advancing theoretical research in scientific disciplines, particularly in theoretical Physics. Easier visualization can help policy makers, investors and economists describe their models without having to resort to lengthy formulaic representations. They can help hedge fund managers identify early trends, Central Bankers design intervention tools, policy makers detect regime changes, among many other applications. As it relates to global macro trading, they can help monitor how financial flows propagate across various investment assets.

In this paper we propose a novel methodology to visualize the complex network of flows in a Time Series system, which we call *Stochastic Flow Diagrams* (SFDs). SFDs are topological representations of a system of equations in differences, such as those encountered in Time Series models. Our method combines elements of Graph Theory and inferential statistics to visualize the structure of a complex system, allowing for an intuitive interpretation of its state and future course. The SFD method takes into consideration the dynamic properties of the system, determining the direction of the flows in terms of lead-lag and causality effects. SFD connectivity is determined by statistical significance of the graph's arcs, which are weighted based on the estimated parameters of the Time Series model. Because SFDs map dynamic systems, they incorporate a time dimension. This is made explicit in the design of the SFD, through the definitions of *outbound arc* and *lagged vertex*.

We have shown that a small number of SFD attributes is able to describe a wide range of Time Series models. In the case of Macroeconomics, SFD allows researchers monitor flow dynamics, as the values adopted by the random variables change over time. One important practical application of this approach is to be able to visualize the status of the system, and anticipate the possibility of overflows. A second application is to simulate the propagation of shocks emanating from alternative vertices, with dramatically different consequences depending on whether the system is in a stable, steady or explosive state. The number of scenarios a researcher can simulate is enormous, and SFDs offer an intuitive way to compute, compare and communicate the alternative outcomes. We refer the reader to Calkin and López de Prado (2014) for a particular application of SFD to the study of Macro financial flows.

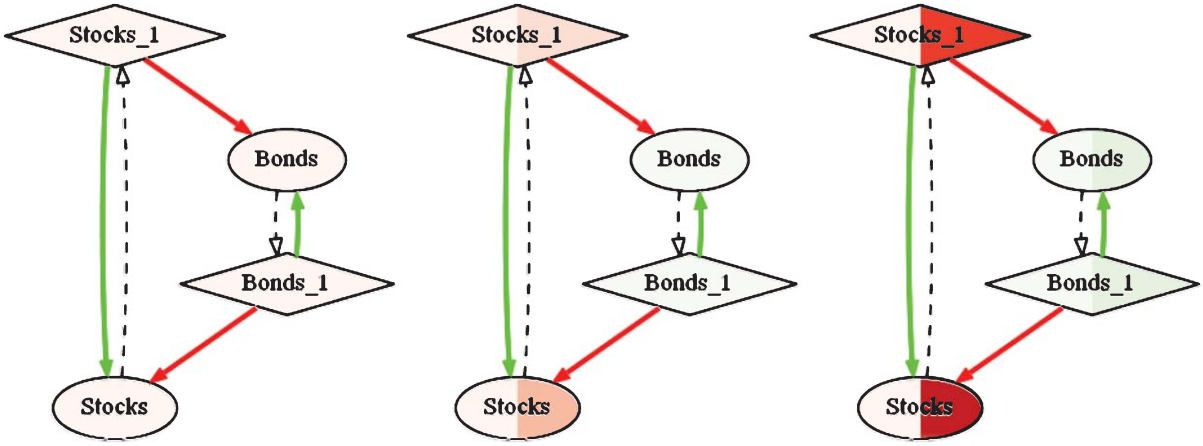

Fig. 10. Evolution of a system in a *Stable* state, after 1, 5 and 10 periods.
Note: In a stable state, the system absorbs the shock and new equilibrium levels are reached after a few periods. The rate of change fades over time, as indicated by the color of the vertices' left half. The cumulative effect converges to the new equilibrium level, as evidenced by the color of the vertices' right half. Note that stocks are much more affected than bonds.

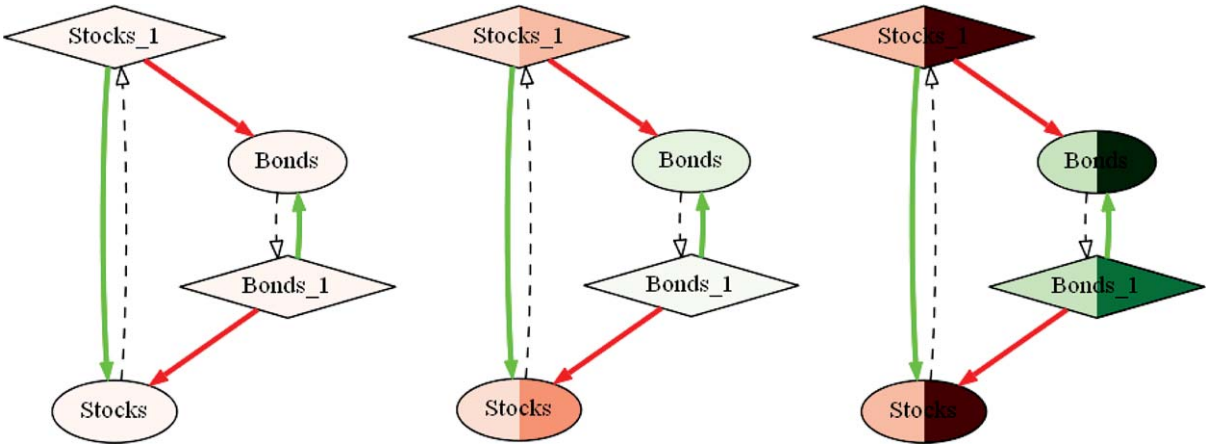

Fig. 11. Evolution of a system in a *Steady* state, after 1, 5 and 10 periods.
Note: In a steady state, the system is unable to absorb the shock, and a drift continues until another shock counters it. The rate of change does not fade over time, as indicated by the color of the vertices' left half. The cumulative effect grows indefinitely, as evidenced by the color of the vertices' right half. Because the drift continues, bonds also end up being significantly affected.

Acknowledgments

We thank the Managing Editor of Algorithmic Finance, as well as three anonymous referees for their suggestions and support. We are grateful to Tony Anagnostakis (Moore Capital), David H. Bailey (Lawrence Berkeley National Laboratory), José Blanco (UBS), Jonathan M. Borwein (University of Newcastle), Peter Carr (Morgan Stanley, NYU), Marco Dion (J.P. Morgan), David Easley (Cornell University), Matthew D. Foreman (University of California, Irvine), Jon Kleinberg (Cornell University), Jeffrey Lange (Guggenheim Partners), Attilio Meucci (KKR, NYU), Alberto Musalem (Federal Reserve Bank of New York), Riccardo Rebonato (PIMCO, University of Oxford) and Luis Viceira (HBS).

Disclaimer

The opinions expressed by the authors of this article do not necessarily reflect the views of Berkeley Lab or Guggenheim Partners. No investment advice of particular course of action is recommended by this article.

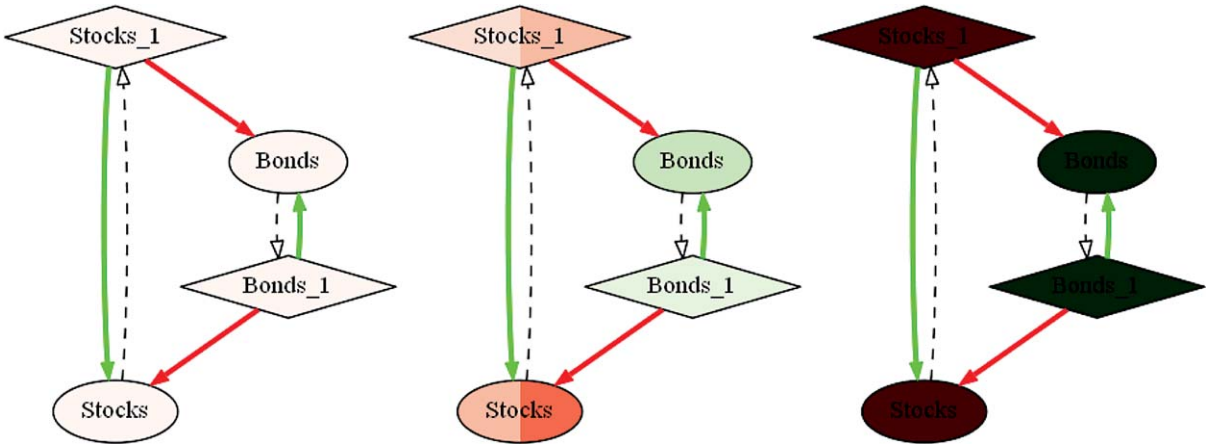

Fig. 12. Evolution of system in an *Explosive* state, after 1, 5 and 10 periods.
Note: In an explosive state, the system magnifies the shock over time. The rate of change increases after each period, as indicated by the color of the vertices' left half. The cumulative value explodes after a few periods, as evidenced by the color of the vertices' right half. The entire system is disrupted and eventually crashes.

Appendices

A.1. Algorithm for SFDs of AR(*p*) processes

Snippet 1 contains the algorithm, coded in Python, that creates the basic SFD architecture of an AR(*p*) process. It makes use of the *Networkx* library, an open-source software developed by scientists at the Los Alamos National Laboratory. Networkx is available at http://networkx.lanl.gov/.

The function is instantiated as *SFD_AR(p)*, where *p* denotes the number of lags. All arcs are then generated, outbound and inbound. Weights are arbitrarily assigned a value of 1. The function returns the resulting weighted digraph as an object *D*. The reader can use this template to remove arcs that are statistically insignificant, and dynamically change the weights.

A.2. Algorithm for SFDs of VAR(p) system on n equations

Snippet 2 contains the algorithm, coded in Python, that creates the basic SFD architecture of a VAR(*p*) system on *n* equations. The function is instantiated as *SFD_VAR(p,n)*, where *p* denotes the number of lags and *n* the number of equations. All arcs are then generated, outbound and inbound. Weights are arbitrarily

```
#!/usr/bin/env python
# On 20131225 by lopezdeprado@lbl.gov
import networkx as nx
#---------------------------------------------------------------------
def SFD_AR(p):
   # Generate AR figures
   D=nx.DiGraph()
   for k in range(p):
      D.add_node('v'+str(k+1),shape='diamond')
      D.add_edge('v'+str(k),'v'+str(k+1),weight=1, \
         label='o'+str(k),style='dashed',arrowhead='onormal') # outbound arc
      D.add_edge('v'+str(k+1),'v0',weight=1, \
         label='a'+str(k+1)) # inbound arc
   return D
```

Snippet 1. *SFD of an AR(p) process.*

assigned a value of 1. The function returns the resulting weighted digraph as an object *D*. Like in the AR(*p*) case, the reader can use this template to remove arcs that are statistically insignificant, and dynamically change the weights.

A.3. Algorithm for SFDs of Svar systems

Snippet 3 contains the algorithm, coded in Python, that creates the basic SFD architecture of a SVAR system, which combines synchronous as well as

```
#!/usr/bin/env python
# On 20131225 by lopezdeprado@lbl.gov
import networkx as nx
#-------------------------------------------------------------------------
def SFD_VAR(p,n):
   # Generate VAR figures
   D=nx.DiGraph()
   range1=range(1,n+1)
   for i in range1:
      for k in range(p):
         D.add_node('v('+str(i)+','+str(k+1)+')',shape='diamond')
         D.add_edge('v('+str(i)+','+str(k)+')', \
            'v('+str(i)+','+str(k+1)+')',weight=1, \
            label='o('+str(i)+','+str(k)+')',style='dashed',arrowhead='onormal') # outbound
         for j in range1:
            D.add_edge('v('+str(i)+','+str(k+1)+')', \
               'v('+str(j)+',0)',weight=1, \
               label='a('+str(i)+','+str(k+1)+','+str(j)+')') # inbound arc
   return D
```

Snippet 2. *SFD of a VAR(p) system on n equations.*

```
#!/usr/bin/env python
# On 20131225 by lopezdeprado@lbl.gov
import networkx as nx
#-------------------------------------------------------------------------
def SFD_SVAR(p,n):
   # Generate SVAR figures
   D=nx.DiGraph()
   range1=range(1,n+1)
   for i in range1:
      range2=range1[:];range2.remove(i)
      for j in range2:
         D.add_edge('v('+str(i)+',0)','v('+str(j)+',0)',weight=1, \
            label='a('+str(i)+',0,'+str(j)+')',arrowhead='inv') # inbound sync arc
      for k in range(p):
         D.add_node('v('+str(i)+','+str(k+1)+')',shape='diamond')
         D.add_edge('v('+str(i)+','+str(k)+')', \
            'v('+str(i)+','+str(k+1)+')',weight=1, \
            label='o('+str(i)+','+str(k)+')',style='dashed',arrowhead='onormal') # outbound
         for j in range1:
            D.add_edge('v('+str(i)+','+str(k+1)+')', \
               'v('+str(j)+',0)',weight=1, \
               label='a('+str(i)+','+str(k+1)+','+str(j)+')') # inbound arc
   return D
```

Snippet 3. *SFD of a SVAR system on n equations and p lags.*

```
#!/usr/bin/env python
# On 20131225 by lopezdeprado@lbl.gov
import networkx as nx, pandas as pd
#---------------------------------------------------------------------
def formGraphFromBetas(betas,aR2):
   # Create a graph using the coefficients matrix as an adjacency matrix
   D,lagNodes=nx.DiGraph(),[]
   # inbound arcs
   for i in betas.index.tolist():
      for j in betas.columns.tolist():
         if betas[j][i]!=0:
            if betas[j][i]<0:color='red'
            else:color='darkgreen'
            width=getWidth(aR2.ix[i][0])
            if '_' in j:
               lagNodes.append((j[:j.find('_')],int(j[j.find('_')+1:])))
               D.add_edge(j,i,weight=betas[j][i],arrowsize=1,color=color, \
                  penwidth=width) # inbound arc
            else:
               D.add_edge(j,i,weight=betas[j][i],arrowsize=1,color=color, \
                  penwidth=width,arrowhead='inv') # inbound sync arc
   # outbound arcs
   if len(lagNodes)>0:
      lagNodes=pd.DataFrame(lagNodes)
      groups=lagNodes.groupby([0]).groups
      for i in groups.keys():
         maxLag=max(lagNodes[1].ix[groups[i]])
         for k in range(0,maxLag):
            if k==0:j=i
            else:j=i+'_'+str(k)
            D.add_node(i+'_'+str(k+1),shape='diamond')
            D.add_edge(j,i+'_'+str(k+1),weight=1,style='dashed', \
               arrowhead='onormal') # outbound arc
   return D
#---------------------------------------------------------------------
def getWidth(value):
   value_=abs(value)/(1+value**2)**.5
   return value_*20+1
```

Snippet 4. *Building the SFD from an a matrix of estimated system parameters.*

lagged regressors. The function is instantiated as *SFD_SVAR(p,n)*, where *p* denotes the number of lags and *n* the number of equations.

A.4. From estimated system parameters to SFDs

Snippet 4 shows how to build a SFD from a matrix of estimated system parameters. This is accomplished by function *formGraphFromBetas(betas)*. The *betas* argument is a Pandas DataFrame object that contains all the estimated betas. Its structure is: One row vector per equation, where each column is associated with one regressor. The column names are the names of the regressors, and the index names are the names of the variable estimated by each equation. The *aR2* argument is another Pandas DataFrame object, containing all the adjusted R-Squares (a single column, with one equation per row). Values in *betas* determine the weight and color of the outbound arcs, while values in aR2 determine their width. See Appendix 6 for additional details regarding the symbology employed.

A.5. SFDs glossary

In this section we will summarize how to read a SFD, and justify the symbology chosen. A small number of attributes is able to represent a wide variety of system architectures. These attributes can be further expanded to signal additional features in complex Time Series systems.

A.5.1. Vertices

Vertices have two shapes:

- **Elliptical**, for a *current variable*. There is only one in AR specifications, and multiple in VAR and SVAR systems.
- **Diamond**, for a *lagged variable*.

Vertices can have two colors:

- **Red:** The variable has a negative value (darker as more negative).
- **Green:** The variable has a positive value (darker as more positive).

If the variable expresses a change in value rather than a cumulative value (e.g., in an equation in differences), then the vertex is divided in two halves, where the left half is colored according to the change in value and the right according to the cumulative value.

The label inside the vertex indicates the variable name, and in the case of a lagged variable, also the order of the lag.

A.5.2. ARCS

Arcs can have three shapes:

- **Dashed arc line, with unfilled arrowhead:** This denotes an *outbound arc*, i.e. an arc that carries flow to a diamond vertex (lagged variable). They appear in all Time Series models. This is a convenient representation, because these arcs can only have a unit weight, hence no colour is needed to convey the weight.
- **Solid arrow, with pointed arrowhead:** This denotes an *inbound arc* involved in a lead-lag effect (connecting a diamond vertex with an elliptical vertex), e.g. in AR processes and VAR systems.
- **Solid arrow, with flat arrowhead:** This denotes the *inbound arc* involved in a contemporaneous effect (connecting two elliptical vertices), e.g. in SVAR systems.

Inbound arcs can have two colors:

- **Red:** Negative weight (darker as more negative).
- **Green:** Positive value (darker as more positive).

The width of an inbound arc is a function of the goodness of fit associated with that equation.

A.6. System Reverberation

Consider a VAR(1) model on 2 variables

$$\begin{aligned} y_{1,t} &= \varphi_{1,1,1} y_{1,t-1} + \varphi_{1,2,1} y_{2,t-1} + \varepsilon_{1,t} \\ y_{2,t} &= \varphi_{2,1,1} y_{1,t-1} + \varphi_{2,2,1} y_{2,t-1} + \varepsilon_{2,t} \end{aligned} \tag{17}$$

where $\varphi_{i,j,k}$ is the regression coefficient associated with equation i, regressor j, after k lags. The error terms satisfy the conditions of a generalized white noise. It is evident that each equation can forecast the variable on the left side of the equality for one step. However, each equation interacts with the other over horizons greater than one, thus forming a system. To see how, let's express this system in matrix form,

$$Y_t = \Phi Y_{t-1} \tag{18}$$

where $Y_t = \begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix}$ and $\Phi = \begin{bmatrix} \varphi_{1,1,1} & \varphi_{1,2,1} \\ \varphi_{2,1,1} & \varphi_{2,2,1} \end{bmatrix}$. The spectral decomposition $\Phi W = W\Lambda$ leads to the eigenvalue equation $|\Phi - I\Lambda| = 0$, where W, Λ are respectively the matrix of eigenvector and eigenvalue of Φ, I the identity matrix, and $|.|$ is the determinant. Operating,

$$\begin{aligned} &\begin{bmatrix} \varphi_{1,1,1} - \Lambda_{1,1} & \varphi_{1,2,1} \\ \varphi_{2,1,1} & \varphi_{2,2,1} - \Lambda_{2,2} \end{bmatrix} = 0 \\ &\Rightarrow (\varphi_{1,1,1} - \Lambda_{1,1})(\varphi_{2,2,1} - \Lambda_{2,2}) - \varphi_{1,2,1}\varphi_{2,1,1} = 0 \end{aligned} \tag{19}$$

This second degree equation has roots in

$$\begin{aligned} \Lambda_{1,1} &= \frac{tr[\Phi] + \sqrt{(tr[\Phi])^2 - 4\,|\Phi|}}{2} \\ \Lambda_{2,2} &= \frac{tr[\Phi] - \sqrt{(tr[\Phi])^2 - 4\,|\Phi|}}{2} \end{aligned} \tag{20}$$

where $tr[\Phi]$ and $|\Phi|$ are respectively the trace and determinant of Φ. Now that we have computed the Λ that makes the matrix $\Phi - I\Lambda$ singular, we can proceed with computing the eigenvector matrix, by finding the kernel of $\Phi - I\Lambda$.

For $\Lambda_{1,1}$, we establish a system of equations

$$\begin{bmatrix} \varphi_{1,1,1} - \Lambda_{1,1} & \varphi_{1,2,1} \\ \varphi_{2,1,1} & \varphi_{2,2,1} - \Lambda_{2,2} \end{bmatrix} \begin{bmatrix} W_{1,1} \\ W_{2,1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (21)$$

Elemental row operations yield the result $\begin{bmatrix} 1 & \frac{\varphi_{1,2,1}}{\varphi_{1,1,1} - \Lambda_{1,1}} \\ 0 & 1 \end{bmatrix}$, so we can reduce the system to

$$W_{1,1} + W_{2,1}\frac{\varphi_{1,2,1}}{\varphi_{1,1,1} - \Lambda_{1,1}} = 0$$
$$W_{2,1} = 1 \quad (22)$$

Applying the especial solutions on the kernel allow us to conclude that

$$W = \begin{bmatrix} \frac{-\varphi_{1,2,1}}{\varphi_{1,1,1} - \Lambda_{1,1}} & \frac{-\varphi_{1,2,1}}{\varphi_{1,1,1} - \Lambda_{2,2}} \\ 1 & 1 \end{bmatrix} \quad (23)$$

Given some initial conditions $y_{1,0}$, $y_{2,0}$, it must be satisfied that $\begin{bmatrix} y_{1,0} \\ y_{2,0} \end{bmatrix} = Wc$, where c is a column vector that solves W for the initial conditions. Assuming that Φ is invertible, we know that Φ must be diagonalizable as $\Phi = W\Lambda W^{-1}$. Then, $\begin{bmatrix} y_{1,1} \\ y_{2,1} \end{bmatrix} = \Phi \begin{bmatrix} y_{1,0} \\ y_{2,0} \end{bmatrix} = W\Lambda W^{-1}Wc = W\Lambda c$. Multiplying k times by Φ will yield $\begin{bmatrix} y_{1,k} \\ y_{2,k} \end{bmatrix} = \Phi^k \begin{bmatrix} y_{1,0} \\ y_{2,0} \end{bmatrix} = W\Lambda^k c$, or what is the same,

$$Y_k = \begin{bmatrix} y_{1,k} \\ y_{2,k} \end{bmatrix} = c_1\Lambda_{1,1}^k \begin{bmatrix} W_{1,1} \\ W_{2,1} \end{bmatrix} + c_2\Lambda_{2,2}^k \begin{bmatrix} W_{1,2} \\ W_{2,2} \end{bmatrix} \quad (24)$$

At the initial conditions, $Y_0 = \begin{bmatrix} y_{1,0} \\ y_{2,0} \end{bmatrix} = c_1 \begin{bmatrix} W_{1,1} \\ W_{2,1} \end{bmatrix} + c_2 \begin{bmatrix} W_{1,2} \\ W_{2,2} \end{bmatrix}$, from where we obtain

$$c_1 = y_{2,0} - c_2$$
$$c_2 = \frac{y_{1,0} - y_{2,0}W_{1,1}}{W_{1,2} - W_{1,1}} \quad (25)$$

Our solution is analytical and in closed-form. This means that the long-run forecast can be estimated in a single calculation for any horizon, without requiring a sequential estimation over a sufficiently large number of iterations.

Now that we know how to estimate this dynamic system, we would like to understand what causes a crash from a mathematical standpoint. Looking at Eq. (24), we can identify the following states:

- *Stable state*: $\left|\Lambda_{i,i}\right| < 1$ for $i = 1, 2$. Both eigenvalues must be smaller than one in absolute value. If imaginary eigenvalues exist, their real part must be smaller than one in absolute value.
- *Steady state*: $\exists i,\ j \left|\left|\Lambda_{i,i}\right| = 1,\ \left|\Lambda_{j,j}\right| < 1$. The absolute value of one eigenvalue is equal to one, and the other is not greater than one in absolute value (or their real part, being imaginary).
- *Explosive state*: $\exists i \left|\left|\Lambda_{i,i}\right| > 1$. The absolute value of any eigenvalue is greater than one (or their real part, being imaginary).

References

Bollobás, B., 2013. Modern Graph Theory, Springer.

Bondy, J., Murty, U., 1976. Graph Theory with Applications, Elsevier Science.

Brualdi, R., 2010. The Mutually Beneficial Relationship of Graphs and Matrices. Conference Board of the Mathematical Sciences, Regional Conference Series in Mathematics, Nr. 115.

Calkin, N., López de Prado, M., 2014. The Topology of Macro Financial Flows: An Application of Stochastic Flow Diagrams. Working paper. Available in SSRN. http://ssrn.com/abstract=2379319

Campbell, J., Lo A., MacKinlay, A., 1996. The Econometrics of Financial Markets, Princeton University Press.

Durrett, R., 2007. Random Graph Dynamics, Cambridge University Press.

Easley, D., Kleinberg, J., 2010. Networks, Crowds and Markets: Reasoning about a Highly Connected World. Cambridge University Press.

Greene, W., 2008. Econometric Analysis, Prentice Hall, 6th Edition.

Hamilton, J., 1994. Time Series Analysis. Princeton University Press.

Harvey, A.C., 1989. Trends, cycles and autoregressions, The Economic Journal 107, pp. 192–201.

Jackson, M., 2010. Social and Economic Networks, Princeton University Press.

Kaiser, David., 2005. Physics and Feynman's diagrams, American Scientist 93, pp. 156–165.

Leontief, W., 1982. Academic economics, Science Magazine, pp. 104–107.

Muirhead, R., 1982. Aspects of Multivariate Statistical Theory. John Wiley & Sons, New York.

Rebonato, R., Denev, A., 2014. Portfolio Management under Stress. Cambridge University Press. 1st Edition.

Tsay, R., 2010. Analysis of Financial Time Series, Wiley, 3rd Edition.

Zellner, A., 1962. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias, Journal of the American Statistical Association 57, pp. 348–368.