Optimization and Engineering
https://doi.org/10.1007/s11081-022-09737-0

RESEARCH ARTICLE

# Multi-period liability clearing via convex optimal control

Shane Barratt[1] · Stephen Boyd[1]

Received: 24 March 2022 / Revised: 11 June 2022 / Accepted: 11 June 2022

**Abstract**
We consider the problem of determining a sequence of payments among a set of entities that clear (if possible) the liabilities among them. We formulate this as an optimal control problem, which is convex when the objective function is, and therefore readily solved. For this optimal control problem, we give a number of useful and interesting convex costs and constraints that can be combined in any way for different applications. We describe a number of extensions, for example to handle unknown changes in cash and liabilities, to allow bailouts, to find the minimum time to clear the liabilities, or to minimize the number of non-cleared liabilities, when fully clearing the liabilities is impossible.

**Keywords** Liability clearing · Convex optimization · Optimal control

## 1 Introduction

Large, complex networks of liabilities are the foundation of modern financial systems. According to the FDIC, there were on the order of five thousand FDIC-insured banks in the United States at the end of 2019 (Federal Deposit Insurance Corporation 2019). Each of these banks owe each other money as a result of bank transfers, loans, and securities issued. Inter-bank settlement is handled today by simple payment systems like Fedwire and CHIPS (Bekaert and Hodrick 2012, Sect. 2.4). Another example of complex liability networks are derivatives exchanges and brokerages, where there are liabilities between clients in the form of derivatives contracts or borrowed shares. A goal shared by all of the entities in these systems is to clear or remove liabilities, which reduces risk and complexity. Each system has its own goals and constraints in its mission to clear liabilities, which must be accounted for.

✉ Stephen Boyd
boyd@stanford.edu

1 Stanford University, Stanford, USA

We consider the general problem of liability clearing, which is to determine a sequence of payments between a set of financial entities to clear (if possible) the liabilities among them over a finite time horizon. We first observe that the dynamics in liability clearing are linear, and describe methods that can be used to remove cycles of liability before any payments are made. We then formulate liability clearing as an optimal control problem with convex objectives and constraints, where the system's state is the cash held by each entity and the liabilities between each entity, and the input to the system is the payments made by each entity to other entities. This formulation has several benefits. First, we can naturally incorporate the goals and constraints in liability clearing in the stage cost function of our optimal control problem. Second, we can efficiently (globally) solve the problem since it is convex. Third, domain specific languages for convex optimization make it easy to prototype new liability clearing mechanisms.

We also extend our formulation to the case where there are exogenous unknown inputs to the dynamics, which represent uncertain future liabilities or cash flows. We propose a solution method based on model predictive control, or shrinking horizon control, which, at each time step, predicts the future unknown inputs, plans a sequences of payments, and then uses just the first of those payments for the next time period. We then illustrate our method on several simulated numerical examples, comparing to a simple pro-rata payment baseline. At the end of the paper, we discuss extensions and variations of our problem, *e.g.*, allowing bailouts, finding the minimum time to clear a set of liabilities, non-time-separable costs, infinite time liability control, and how to minimize the number of non-cleared liabilities.

**Outline** In Sect. 2 we discuss related work as well as its relation to our paper. In Sect. 3 we set out our notation, and describe the dynamics equations and constraints. In Sect. 4 we formulate liability clearing as a convex optimal control problem, and describe a number of useful and interesting convex costs and constraints. In Sect. 5 we extend the optimal control formulation to the case where the dynamics are subject to additional uncontrollable exogenous terms, and propose a standard method called model predictive control for this problem. In Sect. 6 we illustrate the methods described in this paper by applying them to several numerical examples. In Sect. 7 we conclude with a number of extensions and variations on our formulation.

## 2 Related work

The liability clearing problem was originally proposed by Eisenberg and Noe in 2001 (Eisenberg and Noe 2001). Their formulation involves determining a single set of payments to be made between the entities, in contrast to ours, which assumes a sequence of payments are made. Their formulation assumes that these payments can be financed immediately by payments received, whereas we make the realistic financing constraint that entities cannot pay other entities more than the cash they have on hand. (This means that it may take multiple steps to clear liabilities.) In this way, our formulation can be viewed as a supply chain with cash as the commodity (Boyd and Vandenberghe 2018, Sect. 9.5), while their formulation can be viewed as a network flow problem (Harris and Ross 1955), where cash can travel multiple steps through the network at once.

They also make the assumption that all liabilities have equal priority (Eisenberg and Noe 2001, Sect. 2.2), *i.e.*, each entity makes payments proportional to its liabilities to the other entities (we call this a pro-rata constraint; see Sect. 4.2). This means that instead of choosing a matrix of payments between all the entities, they only need to choose a vector of payments made by the entities; the payments are then distributed according to the proportion of liability (they call this vector the clearing vector). Our formulation can include the constraint that payments are made in proportion to liability, but we observe that enforcing this constraint at each step is not the most efficient strategy for clearing liabilities (see Sect. 4.5). Because we can incorporate arbitrary (convex) costs and constraints, our formulation is more flexible and realistic.

Eisenberg and Noe's original liability clearing formulation has been extended in multiple ways to include, *e.g.*, default costs and rescue (Rogers and Veraart 2013), cross-holdings (Elsinger 2009, Sect. 2), claims of different seniority (Elsinger 2009, Sect. 6), fire sales (Cifuentes et al. 2005), multiple assets (Feinstein 2019), and has been used to answer fundamental questions about contagion in and shocks to large financial networks (Glasserman and Young 2015; Feinstein et al. 2017). The extension of our methods to a couple of these cases is described in Sect. 7. The sensitivity of clearing vectors to liabilities has also been analyzed, using implicit differentiation (Feinstein et al. 2018) and the Farkas lemma (Khabazian and Peng 2019). Using the techniques of differentiable convex optimization solution maps (Agrawal et al. 2019), we can perform similar sensitivity analysis of our liability control problems. Another tangential but related problem is modeling of liquidity risk and funding runs; of particular note here are the Diamond and Dybvig model of bank runs (Diamond and Dybvig 1983) and the Allen and Gale model of interbank lending (Allen and Gale 2000) (see, *e.g.*, Glasserman and Young (2015, Sect. 4) for a survey).

Eisenberg and Noe's formulation has also been extended to multiple periods. Capponi and Chen proposed a multi-period clearing framework with a lender of last resort (*i.e.*, bailouts, see Sect. 7.1) and exogenous liabilities and cash flows (see Sect. 5.2), and proposed a number of heuristic policies for controlling risk (Capponi and Chen 2015). However, their formulation does not include the financing constraint, meaning liabilities can be (if possible) cleared in one step; their focus is more on cash injection and defaults. Other related works include an extension to continuous time (Banerjee et al. 2018), incorporation of multiple maturities and insolvency law (Kusnetsov and Veraart 2019), incorporation of contingent payments (Banerjee and Feinstein 2019), and an infinite-time treatment (Bardoscia et al. 2019).

## 3 Notation and dynamics

In this section we set out our notation, and describe the dynamics equations and constraints.

**Entities and cash held** We consider a financial system with $n$ financial entities or agents, such as banks, which we refer to as entities $1, \ldots, n$. These entities make payments to each other over discrete time periods $t = 1, \ldots, T$, where $T$ is the time horizon. The time periods could be any length of time, *e.g.*, each time period could represent a business day. We let $c_t \in \mathbf{R}^n_+$ denote the cash held by each of the entities, with $(c_t)_i$ being the amount held by entity $i$ in dollars at time period $t$. If the entities are banks, then the cash held is the bank's reserves, *i.e.*, physical cash and deposits at the central bank. If the entities are individuals or corporations, then the cash held is the amount of deposits at their bank.

**Liability matrix** Each entity has liabilities or obligations to the other entities, which represent promised future payments. We represent these liabilities by the *liability matrix* $L_t \in \mathbf{R}^{n\times n}_+$, where, at time period $t$, $(L_t)_{ij}$ is the amount in dollars that entity $i$ owes entity $j$ (Eisenberg and Noe 2001, Sect. 2.2). We will assume that $(L_t)_{ii} = 0$, *i.e.*, the entities do not owe anything to themselves. Note that $L_t\mathbf{1} \in \mathbf{R}^n_+$ is the vector of total liabilities of the entities, *i.e.*, $(L_t\mathbf{1})_i$ is the total amount that entity $i$ owes the other entities, in time period $t$, where $\mathbf{1}$ is the vector with all entries one. Similarly, $L_t^T\mathbf{1} \in \mathbf{R}^n_+$ is the vector of total amounts owed to the entities by others, *i.e.*, $(L_t^T\mathbf{1})_i$ is the total amount owed to entity $i$ by the others. The *net liability* of the entities at time period $t$ is $L_t\mathbf{1} - L_t^T\mathbf{1}$, *i.e.*, $(L_t\mathbf{1} - L_t^T\mathbf{1})_i$ is the net liability of entity $i$. When $(L_t)_{ij} = 0$, we say that the liability between entity $i$ and $j$ is *cleared* (in time period $t$). The scalar quantity $\mathbf{1}^T L_t \mathbf{1}$ is the *total gross liability* between all the entities. When it is zero, which occurs only when $L_t = 0$, all liabilities between the entities have been cleared.

**Payment matrix** At each time step, each entity makes cash payments to other entities. We represent these payments by the *payment matrix* $P_t \in \mathbf{R}^{n\times n}_+$, $t = 1, \ldots, T-1$, where $(P_t)_{ij}$ is the amount in dollars that entity $i$ pays entity $j$ in time period $t$. We assume that $(P_t)_{ii} = 0$, *i.e.*, entities do not pay themselves. Thus $P_t\mathbf{1} \in \mathbf{R}^n_+$ is the vector of total payments made by the entities to others in time period $t$, *i.e.*, $(P_t\mathbf{1})_i$ is the total cash paid by entity $i$ to the others. The vector $P_t^T\mathbf{1} \in \mathbf{R}^n_+$ is the vector of total payments received by the entities from others in time period $t$, *i.e.*, $(P_t^T\mathbf{1})_i$ is the total payment received by entity $i$ from the others. Each entity can pay others no more than the cash that it has on hand, so we have the constraint

$$P_t\mathbf{1} \le c_t, \quad t = 1, \ldots, T-1, \tag{1}$$

where the inequality is meant elementwise.

**Dynamics** The liability and cash follow the linear dynamics

$$L_{t+1} = L_t - P_t, \quad t = 1, \ldots, T-1, \tag{2}$$

$$c_{t+1} = c_t - P_t\mathbf{1} + P_t^T\mathbf{1}, \quad t = 1, \dots, T-1. \tag{3}$$

The first equation says that the liability is reduced by the payments made, and the second says that the cash is reduced by the total payments and increased by the total payments received.

These dynamics can be extended to include an interest rate for cash, as well as additional cash flows into and out of the entities, and additional liabilities among the entities. For simplicity, we continue with the simple dynamics (2) and (3) above, and describe some of these extensions in Sect. 7.

**Monotonicity of liabilities** Since $L_t \geq 0$, these dynamics imply that

$$P_t \leq L_t, \quad t = 1, \dots, T-1, \tag{4}$$

*i.e.*, each entity cannot pay another entity more than its liability. We also observe that

$$L_{t+1} \leq L_t, \quad t = 1, \dots, T-1, \tag{5}$$

where the inequality is elementwise, which means that each liability is non-increasing in time. We conclude that if the liability of entity $i$ to entity $j$ is cleared in time period $t$, it will remain cleared for all future time periods. In other words, the sparsity pattern (*i.e.*, which entries are nonzero) of $L_t$ cannot increase over time. The inequality (4) implies that once a liability between entries has cleared, no further payments will be made. This tells us that the sparsity patterns of $P_t$ and $L_t$ are no larger than the sparsity pattern of $L_1$.

**Net worth** The *net worth* of each entity at the beginning of time period $t$ is the cash it holds minus the total amount it owes others, plus the total amount owed to it by others, or

$$w_t = c_t - L_t\mathbf{1} + L_t^T\mathbf{1},$$

where $w_t \in \mathbf{R}^n_+$ is the vector of net worth of the entities. (The second and third terms are the negative net liability.) The net worth is an invariant under the dynamics, since

$$\begin{aligned} w_{t+1} &= c_{t+1} - L_{t+1}\mathbf{1} + L_{t+1}^T\mathbf{1}, \\ &= c_t - P_t\mathbf{1} + P_t^T\mathbf{1} - (L_t - P_t)\mathbf{1} + (L_t - P_t)^T\mathbf{1}, \\ &= c_t - L_t\mathbf{1} + L_t^T\mathbf{1}, \\ &= w_t. \end{aligned}$$

**Default** If $(w_1)_i < 0$, *i.e.*, the initial net worth of entity $i$ is negative, then it will have to default; it cannot reduce its net liability to zero. If an entity defaults, then it will find itself unable to fully pay the entities it owes money to, which might cause those entities to default as well. Such a situation is called a *default cascade* (Eisenberg and Noe 2001, Sect. 2.4).

## 3.1 Liability cycle removal

**Graph interpretation** The liabilities between the entities can be interpreted as a weighted directed graph, where the nodes represent the entities, and the directed edges represent liabilities between entities, with weights given by the liabilities. In this interpretation, the liability matrix is simply the weighted adjacency matrix.

**Liability cycle removal** Some of the liabilities between entities can be reduced or removed without the need to make payments between them. This happens when there are one or more *liability cycles*. A liability cycle is a cycle in the graph described above, or a sequence of positive liabilities that starts and ends at the same entity and does not visit an entity more than once. If there is a liability cycle, then each liability in the cycle can be reduced by the smallest liability present in the cycle, which reduces at least one of the liabilities in the cycle to zero (which therefore breaks the cycle). Removing a liability cycle in this manner keeps the net liabilities of each entity, $L_t\mathbf{1} - L_t^T\mathbf{1}$, constant. The simplest case occurs with a cycle of length two: If $(L_t)_{ij}$ and $(L_t)_{ji}$ are both positive, *i.e.*, entities $i$ and $j$ each owe the other some positive amount, then we can replace these liabilities with

$$(L_t)_{ij} - \min\{(L_t)_{ij}, (L_t)_{ji}\}, \qquad (L_t)_{ji} - \min\{(L_t)_{ij}, (L_t)_{ji}\},$$

which will reduce one of the two liabilities (the one that was originally smaller) to zero.

Given a liability matrix $L$, we give two ways to remove liability cycles, a greedy algorithm and a formulation of the problem as a linear program. This problem is referred to in the literature as portfolio compression (D'Errico and Roukny 2017; Schuldenzucker and Seuken 2019; Veraart 2019; Amini and Feinstein 2020) and payment netting (Shapiro 1978; O'Kane 2014, 2017).

**Greedy cycle clearing algorithm** The greedy cycle clearing algorithm begins by searching for a liability cycle, which can be done using a topological sort (Kahn 1962). If there are no liability cycles, the algorithm terminates. On the other hand, if there is a liability cycle, the algorithm reduces each liability in the cycle by the smallest liability present in the cycle, thus removing the cycle. This process is repeated until there are no more liability cycles. This algorithm was first proposed in 2009 in a patent filed by TriOptima (Brouwer 2009), a portfolio compression company owned by the CME group that has reported clearing over 1000 trillion dollars of liabilities through 2017.

**Optimal cycle clearing via linear programming** The greedy algorithm described above can be improved upon if our goal is not to just remove cycles, but also to remove as much total gross liability as possible. The problem is to find a new liability matrix $\tilde{L} \leq L$ with the smallest total gross liability, subject to the constraint that the net liabilities remains the same. This can be accomplished by solving the linear program

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T \tilde{L} \mathbf{1} \\ \text{subject to} & L\mathbf{1} - L^T\mathbf{1} = \tilde{L}\mathbf{1} - \tilde{L}^T\mathbf{1}, \\ & 0 \le \tilde{L} \le L. \end{array} \tag{6}$$

with variable $\tilde{L}$.

To the best knowledge of the authors, the linear programming formulation of this problem was first proposed by Shapiro (1978). In his formulation, he incorporated transaction costs by making the objective $\mathbf{Tr}(C^T(L - \tilde{L}))$ for a given transaction cost matrix $C \in \mathbf{R}_+^{n \times n}$. Other objectives are possible, *e.g.*, the sum of the squared liabilities (O'Kane 2014, Sect. 3.3).

Liability cycle removal could be carried out before the payments have begun. From (5), this implies that no cycles would appear; that is, $L_t$ would contain no cycles for $t = 1, \dots, T$. We note however that the methods described in this paper work regardless of whether there are cycles, or whether liability cycle removal has been carried out; that is, liability cycle removal is optional for the methods described in this paper.

## 4 Liability control

### 4.1 Optimal control formulation

We now formulate the problem of finding a suitable sequence of payments that clear (or at least reduce) the liabilities among the entities as a convex optimal control problem. Given an initial liability matrix $L^{\text{init}}$ and cash $c^{\text{init}}$, the *liability control problem* is to choose a sequence of payments $P_1, \dots, P_{T-1}$ so as to minimize a sum of stage costs,

$$\sum_{t=1}^{T-1} g_t(c_t, L_t, P_t) + g_T(c_T, L_T), \tag{7}$$

where the function $g_t : \mathbf{R}_+^n \times \mathbf{R}_+^{n \times n} \times \mathbf{R}_+^{n \times n} \to \mathbf{R} \cup \{+\infty\}$ is the (possibly time-varying) stage cost, and $g_T : \mathbf{R}_+^n \times \mathbf{R}_+^{n \times n} \to \mathbf{R} \cup \{+\infty\}$ is the terminal stage cost.

Infinite values of the stage cost $g_t$ (or $g_T$) are used to express constraints on $c_t$, $L_t$, or $P_t$. To impose the constraint $(c_t, L_t, P_t) \in \mathcal{C}_t$, we define $g_t(c_t, L_t, P_t) = +\infty$ for $(c_t, L_t, P_t) \notin \mathcal{C}_t$. As a simple example, the final stage cost

$$g_T(c_T, L_T) = \begin{cases} 0 & L_T = 0, \\ \infty & L_T \neq 0, \end{cases}$$

imposes the constraint that the sequence of payments must result in all liabilities cleared at the end of the time horizon. Here $g_T$ is the *indicator function* of the constraint $L_T = 0$. (The indicator function of a constraint has the value 0 when the constraint is satisfied, and $+\infty$ when it is violated.)

The liability control problem has the form

$$\begin{array}{ll} \text{minimize} & \sum_{t=1}^{T-1} g_t(c_t, L_t, P_t) + g_T(c_T, L_T) \\ \text{subject to} & L_{t+1} = L_t - P_t, \quad t = 1, \dots, T-1, \\ & c_{t+1} = c_t - P_t\mathbf{1} + P_t^T\mathbf{1}, \quad t = 1, \dots, T-1, \\ & P_t\mathbf{1} \le c_t, \quad t = 1, \dots, T-1, \\ & P_t \ge 0, \quad t = 1, \dots, T-1, \\ & L_t \ge 0, \quad t = 1, \dots, T, \\ & c_1 = c^{\text{init}}, \quad L_1 = L^{\text{init}}, \quad c_T \ge 0, \end{array} \tag{8}$$

with variables $c_t$, $L_t$, $t = 1, \dots, T$, and $P_t$, $t = 1, \dots, T-1$. (The constraint $c_t \ge 0$ is implied by $P_t\mathbf{1} \le c_t$.) We refer to this as the *liability clearing control problem.* It is specified by the stage cost functions $g_1, \dots, g_T$, the initial liability matrix $L^{\text{init}}$, and the initial cash vector $c^{\text{init}}$. We observe that the last four sets of inequality constraints could be absorbed into the stage cost functions $g_t$ and $g_T$; for clarity we include them in (8) explicitly.

**Convexity** We will make the assumption that the stage cost functions $g_t$ and $g_T$ are convex, which implies that the liability control problem (8) is a convex optimization problem (Boyd and Vandenberghe 2004). This implies that it can be (globally) solved efficiently, even at large scale; this is discussed further in Sect. 4.4. Perhaps more important from a practical point of view is that it can be solved with near total reliability, with no human intervention, and at high speed if needed.

We make the assumption not just because of the computational advantages that convexity confers, but also because there are very reasonable choices of the cost functions that satisfy the convexity assumption. It is also true that some reasonable cost functions are not convex; we give an example in Sect. 7.5.

### 4.2 Constraints

In this section we describe some examples of useful constraints, which can be combined with each other or any of the cost functions described below. They are all convex.

**Liability clearance** We can constrain the liabilities to be fully cleared at time $T$ with the constraint

$$L_T = 0.$$

If $w_1 \ngeq 0$ or the liabilities cannot be cleared in time, the liability clearing problem (8) with this constraint will be infeasible.

**Pro-rata constraint** The proportional liability of each entity is the proportion of its total liability that it owes to the other entities, which for entity $i$ is $(L_t)_i/(L_t\mathbf{1})_i$. We can constrain the final proportional liability of each entity to be equal to the initial proportional liability with the linear constraint

$$\mathbf{diag}(L_1\mathbf{1})L_T = \mathbf{diag}(L_T\mathbf{1})L_1, \tag{9}$$

where $\mathbf{diag}(x)$ is the diagonal matrix with $x$ on its diagonal. This constraint also holds if $L_T = 0$, *i.e.*, the sequence of payments clears all liabilities.

**Cash minimums** Cash minimums, represented by the vector $c^{\min} \in \mathbf{R}^n_+$, where $(c^{\min})_i$ is the minimum cash that the entity $i$ is allowed to hold, can be enforced with the constraint

$$c_t \geq c^{\min}, \quad t = 1, \dots, T.$$

Cash minimums can arise for a number of reasons, one of them being reserve requirements for banks (Board of Governors 2020).

**Payment maximums** We can constrain the payment between entities to be below some maximum payment $P^{\max} \in \mathbf{R}^{n\times n}_+$, where $(P^{\max})_{ij}$ is the maximum allowable payment from entity $i$ to entity $j$, with the constraint

$$P_t \leq P^{\max}, \quad t = 1, \dots, T-1.$$

We can impose a limit on how much cash each entity uses for payments with the constraint

$$P_t\mathbf{1} \leq \beta c_t, \quad t = 1, \dots, T-1,$$

where $0 < \beta \leq 1$ is the fraction of the entity's cash that can be used to make payments in each time period.

**Payment deadlines** Deadlines on payments are represented by the set

$$\Omega \subseteq \{1, \dots, T\} \times \{1, \dots, n\} \times \{1, \dots, n\}.$$

If $(t, i, j) \in \Omega$, we require that the liability between entities $i$ and $j$ becomes zero at time $t$. This results in the constraints $(L_t)_{ij} = 0$ for all $(t, i, j) \in \Omega$.

**Progress milestones** We can impose the constraint that the liabilities are reduced by the fraction $\eta \in (0, 1)$ in $\tau$ time periods, with $L_\tau \leq \eta L_1$.

### 4.3 Costs

In this section we list some interesting and useful convex stage costs. We note that any combination of the constraints above can be included with any combination of the costs listed below, by adding their indicator functions to the cost.

**Weighted total gross liability** A simple and useful stage cost is a weighted total gross liability,

$$g_t(c_t, L_t, P_t) = \mathbf{Tr}(C^T L_t) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij}(L_t)_{ij}, \tag{10}$$

where the matrix $C \in \mathbf{R}_{+}^{n \times n}$ represents the (marginal) cost of each liability. When $C = \mathbf{1}\mathbf{1}^T$ (*i.e.*, $C_{ij} = 1$ for all $i$ and $j$), this stage cost is simply the total gross liability $\mathbf{1}^T L_t \mathbf{1}$ at time $t$. When $C$ is not the all ones matrix, it encourages reducing liabilities $L_{ij}$ with higher weights $C_{ij}$.

**Total squared gross payment** Another simple and useful stage cost is the total squared gross payment,

$$g_t(c_t, L_t, P_t) = \mathbf{Tr}(D^T P_t^2) = \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij}(P_t)_{ij}^2,$$

where $D \in \mathbf{R}_{+}^{n \times n}$ represents the cost of each squared payment, and the square is taken elementwise. This stage cost is meant to reduce the size of payments made between entities. As a result of the super-linearity of the square function, it is more sensitive to large payments between the entities than smaller ones. In control terms, the sum of squared payments is our *control effort*, which we would like to be small. It is a traditional term in optimal control.

**Distance from cash to net worth** If the liability is cleared, *i.e.*, $L_t = 0$, then the cash held by each entity will be equal to its net worth, or $c_t = w_t$. We can penalize the distance from the cash held by each entity to its net worth with, *e.g.*, the cost function

$$g_t(c_t, L_t, P_t) = \|c_t - w_1\|_2^2.$$

If we want to make $c_t$ exactly equal to $w_1$ in as many entries as possible as quickly as possible, we can replace the cost above with the $\ell_1$ norm $\|c_t - w_1\|_1$.

**Time-weighted stage cost** Any of these stage costs can be time-weighted. That is, if the stage cost is time-invariant, *i.e.*, $g_t = g$ for some stage cost $g$, the time-weighted stage cost is

$$g_t(c_t, L_t, P_t) = \gamma^{t-1} g(c_t, L_t, P_t),$$

where $\gamma > 0$. For $\gamma > 1$, this stage cost preferentially rewards the stage cost being decreased later (*i.e.*, for large $t$); for $\gamma < 1$, it represents a traditional discount factor, which preferentially rewards the stage cost being decreased earlier (*i.e.*, for small $t$). With $\gamma = 1$, we treat stage costs at different time periods the same.

### 4.4 Computational efficiency

Since problem (8) is a convex optimization problem, it can be solved efficiently (Boyd and Vandenberghe 2004), even for very large problem sizes. The number of variables and constraints in the problem is on the order $Tn^2$. However, this convex optimization problem is often very sparse. The inequalities (4) and (5) imply that $L_t$ and $P_t$ can only have nonzero entries where $L^{\mathrm{init}}$ does. This means that the number of variables can be reduced to order $T\mathbf{nnz}(L^{\mathrm{init}})$ variables, where $\mathbf{nnz}(L^{\mathrm{init}})$ is the number of nonzero entries in the initial liability matrix. (In Appendix 1, we give an alternative formulation of the liability clearing control problem that exploits this sparsity preserving property.) Due to the block-banded nature of the optimal control problem, the computational complexity grows linearly in $T$; see, *e.g.*, (Boyd and Vandenberghe 2004, Sect. A.3).

As a practical matter, we can easily solve the liability clearing problem with $n = 1000$ entities, $\mathbf{nnz}(L^{\mathrm{init}}) = 5000$, and $T = 20$, using generic methods running on an Intel i7-8700K CPU, in under a minute. Small problems, with say $n = 10$ entities, $\mathbf{nnz}(L^{\mathrm{init}}) = 30$, and $T = 20$ can be solved in under a millisecond, using techniques of code generation such as CVXGEN (Mattingley and Boyd 2009, 2012).

It is very easy to express the liability clearing control problem using domain specific languages for convex optimization, such as CVX (Grant and Boyd 2008, 2014), YALMIP (Lofberg 2004), CVXPY (Diamond and Boyd 2016; Agrawal et al. 2018), Convex.jl (Udell et al. 2014), and CVXR (Fu 2019). These languages make it easy to rapidly prototype and experiment with different cost functions and constraints.In each of these languages, the liability control problem can be specified in just a few tens of lines of very clear and transparent code.

### 4.5 Pro-rata baseline method

We describe here a simple and intuitive scheme for determining cash payments $P_1, \dots, P_{T-1}$. We will use this as a baseline method to compare against the optimal control method described above.

The payment $P_t$ is determined as follows. At each time step, each entity pays as much as possible pro-rata, *i.e.*, in proportion to how much it owes the other entities, up to its liability. Define the liability proportion matrix as

$$\Pi = \mathbf{diag}(1/(L^{\mathrm{init}}\mathbf{1}))L^{\mathrm{init}},$$

so $(\Pi)_{ij}$ is the fraction of entity $i$'s total liability that it owes to entity $j$. The pro-rata baseline has the form

$$P_t = \min(\mathbf{diag}(c_t)\Pi, L_t), \tag{11}$$

where min is taken elementwise. We will see that the (seemingly sensible) pro-rata baseline is not an efficient strategy for optimally clearing liabilities.

## 5 Liability control with exogenous unknown inputs

In this section we extend the optimal control formulation in Sect. 4 to handle additional (exogenous) terms in the liability and cash dynamics, unrelated to the clearing process and payments. When these additional terms are known, we obtain a straightforward generalization of the liability clearing control problem, with a few extra terms in the dynamics equations. For the case when they are not known ahead of time, we propose a standard method called model predictive control (MPC), or shrinking horizon control (Bemporad 2006; Rawlings and Mayne 2009; Mattingley et al. 2011). MPC has been used successfully in a wide variety of applications, for example, in supply chain management (Cho et al. 2003), finance (Boyd et al. 2017), automatic control (Falcone et al. 2007; Blackmore et al. 2010), and energy management (Ma et al. 2011; Soltani et al. 2011; Moehle et al. 2019). It has been observed to work well even when the forecasts are not particularly good (Wang and Boyd 2009, Sect. 4).

### 5.1 Optimal control with exogenous inputs

We replace the dynamics equations (2) and (3) with

$$L_{t+1} = L_t - P_t + W_t, \quad t = 1, \dots, T-1, \tag{12}$$

$$c_{t+1} = c_t - P_t\mathbf{1} + P_t^T\mathbf{1} + w_t, \quad t = 1, \dots, T-1, \tag{13}$$

where $W_t \in \mathbf{R}^{n\times n}$ is the liability adjustment at time $t$, and $w_t \in \mathbf{R}^n$ is the exogenous cash flow at time $t$. The liability adjustment $W_t$ can originate from entities creating new liability agreements; the cash flow $w_t$ can originate from payments received or made by an entity, unrelated to clearing liabilities. The terms $W_t$ and $w_t$ are exogenous inputs in our dynamics, *i.e.*, additional terms that affect the liabilities and cash, but are outside our control (at least, for the problem of clearing liabilities). The cash on hand constraint (1) is modified to be

$$P_t\mathbf{1} \le c_t + w_t, \quad t = 1, \dots, T-1, \tag{14}$$

where $c_t + w_t$ is the cash on hand after the exogenous cash flow.

When the exogenous inputs are known (which might occur, for example, when all the exogenous cash flows and liability updates are planned or scheduled), we obtain a straightforward generalization of the liability clearing control problem,

$$\begin{array}{ll}
\text{minimize} & \sum_{t=1}^{T-1} g_t(c_t, L_t, P_t) + g_T(c_T, L_T) \\
\text{subject to} & L_{t+1} = L_t - P_t + W_t, \quad t = 1, \dots, T-1, \\
& c_{t+1} = c_t - P_t\mathbf{1} + P_t^T\mathbf{1} + w_t, \quad t = 1, \dots, T-1, \\
& P_t\mathbf{1} \le c_t + w_t, \quad t = 1, \dots, T-1, \\
& P_t \ge 0, \quad t = 1, \dots, T-1, \\
& c_t \ge 0, \quad L_t \ge 0, \quad t = 1, \dots, T, \\
& c_1 = c^{\text{init}}, \quad L_1 = L^{\text{init}},
\end{array} \tag{15}$$

with variables $c_t$, $L_t$, and $P_t$.

## 5.2 Optimal control with unknown exogenous inputs

We now consider a more common case, where $w_t$ and $W_t$ are not known, or not fully known, when the sequence of payments is chosen. It would be impossible to choose the payment in time period $t$ without knowing $w_t$; otherwise we cannot be sure to satisfy (14). For this reason we assume that $W_t$ and $w_t$ are known at time period $t$, and therefore can be used when we choose the payment $P_t$. (An alternative interpretation is that the exogenous cash arrives before we make payments in period $t$.) Thus at time period $t$, when $P_t$ is chosen, we assume that $w_1, \dots, w_t$ and $W_1, \dots, W_t$ are all known.

**Forecasts.** At time period $t$, we do not know $w_{t+1}, \dots, w_{T-1}$ or $W_{t+1}, \dots, W_{T-1}$. Instead we use *forecasts* of these quantities, which we denote by

$$\hat{w}_{\tau|t}, \quad \hat{W}_{\tau|t}, \quad \tau = t+1, \dots, T-1.$$

We interpret the subscript $\tau|t$ as meaning our forecast of the quantity at time period $\tau$, made at time period $t$. These forecasts can range from sophisticated ones based on machine learning to very simple ones, like $\hat{w}_{\tau|t} = 0$, $\hat{W}_{\tau|t} = 0$, *i.e.*, we predict that there will be no future adjustments to the cash or liabilities. We will take $\hat{w}_{\tau|t} = w_\tau$ and $\hat{W}_{\tau|t} = W_\tau$ for $\tau \leq t$; that is, our 'forecasts' for the current and earlier times are simply the values that were observed.

**Shrinking horizon policy** We now describe a common heuristic for choosing $P_t$ at time period $t$, called MPC. The idea is very simple: we solve the problem (15), over the remaining horizon from time periods $t$ to $T$, replacing the unknown quantities with forecasts. That is, we solve the problem

$$\begin{array}{ll} \text{minimize} & \sum_{\tau=t}^{T-1} g_\tau(c_\tau, L_\tau, P_\tau) + g_T(c_T, L_T) \\ \text{subject to} & L_{\tau+1} = L_\tau - P_\tau + \hat{W}_{\tau|t}, \quad \tau = t, \dots, T-1, \\ & c_{\tau+1} = c_\tau - P_\tau \mathbf{1} + P_\tau^T \mathbf{1} + \hat{w}_{\tau|t}, \quad \tau = t, \dots, T-1, \\ & P_\tau \mathbf{1} \leq c_\tau + w_\tau, \quad \tau = t, \dots, T-1, \\ & P_\tau \geq 0, \quad \tau = t, \dots, T-1, \\ & c_\tau \geq 0, \quad L_\tau \geq 0, \quad \tau = t, \dots, T, \end{array} \tag{16}$$

with variables $c_{t+1}, \dots, c_T$, $L_{t+1}, \dots, L_T$, and $P_t, \dots, P_{T-1}$. In (16), $c_t$ and $L_t$ are known; they are not variables, and we take $\hat{W}_{t|t} = W_t$ and $\hat{w}_{t|t} = w_t$, which are known. We can interpret the solution of (16) as a *plan of action* from time period $t$ to $T$.

We choose $P_t$ as the value of $P_t$ that is a solution of (16). Thus, at time period $t$ we *plan* a sequence of payments (by solving (16)); then we *act* by actually making the payments in the first step of our plan. MPC has been observed to perform well in many applications, even when the forecasts are not particularly good, or simplistic (*e.g.*, zero).

**Pro-rata baseline policy** We observe that the pro-rata baseline payments (11) are readily extended to the case when we have exogenous inputs, with $w_t$ and $W_t$ known at time period $t$. First, we define the liability proportion matrix at time $t$ as

$$\Pi_t = \mathbf{diag}(1/L_t^{\text{run}})L_t^{\text{run}},$$

where $L_t^{\text{run}} = L^{\text{init}} + \sum_{\tau=1}^{t} W_\tau$ is the running sum of liabilities. The pro-rata baseline policy then has the form

$$P_t = \min(\mathbf{diag}(c_t + w_t)\Pi_t, L_t + W_t). \tag{17}$$

## 6 Examples

The code for all of these examples has been made available online at
www.github.com/cvxgrp/multi_period_liability_clearing

We use CVXPY (Diamond and Boyd 2016; Agrawal et al. 2018) to formulate the problems and solve them with MOSEK (2020).

**Initial liability matrix** We use the same initial liability matrix $L^{\text{init}}$ for each example, with $n = 200$ entities. We choose the sparsity pattern of $L^{\text{init}}$ as 2000 random off-diagonal entries (so on average, each entity has an initial liability to 10 others). The nonzero entries of $L^{\text{init}}$ are then sampled independently from a standard log-normal distribution. While we report results below for this one problem instance, numerical experiments with a wide variety of other instances show that the results are qualitatively similar. We note that our example is purely illustrative, and that further experimentation needs to be performed on problem instances that bear more structural similarity to real world financial networks (Boss et al. 2004).

### 6.1 Liability clearing

We consider the problem of clearing liabilities over $T = 10$ time steps, *i.e.*, we have the constraint that the final liabilities are cleared, $L_T = 0$. We set the initial cash to the minimum nonnegative cash required so each entity has nonnegative net worth, or

$$c^{\text{init}} = \max(L^{\text{init}}\mathbf{1} - (L^{\text{init}})^T\mathbf{1}, 0),$$

where max is meant elementwise.

**Total gross liability**

The first stage cost function we consider is

$$g_t(c_t, L_t, P_t) = \mathbf{1}^T L_t \mathbf{1},$$

the total gross liability at each time $t$. We compare the solution to the liability control problem (8) using this stage cost function with the pro-rata baseline method described in Sect. 4.5. The total gross liability and the number of non-cleared

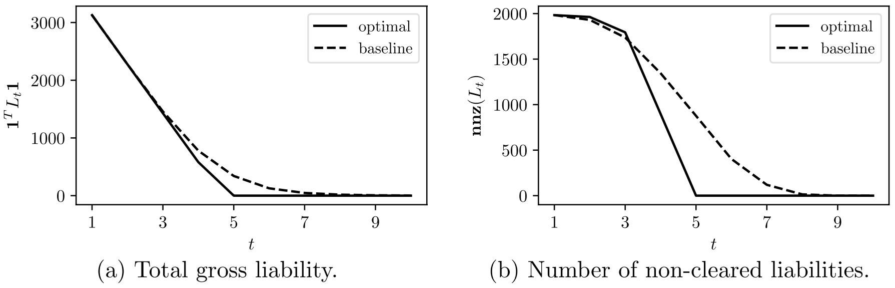


**Fig. 1** Minimizing the sum of total gross liabilities. The solid line is the optimal payment schedule. The dashed line is the pro-rata baseline method

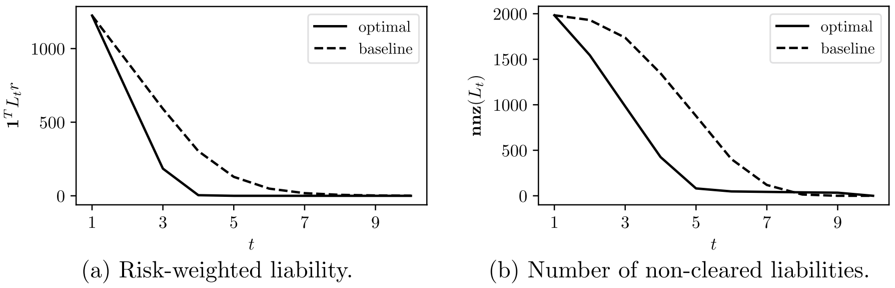


**Fig. 2** Minimizing the sum of risk-weighted liabilities. The solid line is the optimal payment schedule. The dashed line is the pro-rata baseline method

liabilities at each step of both sequences of payments are shown in Fig. 1. The optimal sequence of payments clears the liabilities by $t = 5$, while the baseline clears them by $t = 8$.

## Risk-weighted liability

Suppose we believe that the risk of each entity is proportional to $r = \exp(-w_1)$, where exp is taken elementwise, *i.e.*, higher net worth implies lower risk. A reasonable stage cost function is then risk-weighted liability

$$g_t(c_t, L_t, P_t) = \mathbf{1}^T L_t r.$$

This stage cost encourages clearing the liabilities for high risk entities before low risk entities. We compare the solution to the liability control problem (8) using this stage cost function with the pro-rata baseline method in Sect. 4.5. The total gross liability and the number of non-cleared liabilities at each step of both sequences of payments are shown in Fig. 2. We observe that the liabilities are still cleared by $t = 5$, but the liabilities are much sparser, since the liabilities of high risk entities are cleared before those of low risk entities. We also note that the optimal payment sequence is much faster at reducing risk than the baseline.

**Total squared gross payment**To the total gross liability stage cost above, we add the total squared payments, resulting in the stage cost

$$g_t(c_t, L_t, P_t) = \mathbf{1}^T L_t \mathbf{1} + \lambda \mathbf{1}^T P_t^2 \mathbf{1},$$

where $\lambda > 0$ is a parameter. This choice of stage cost penalizes large payments, and stretches the liability clearing over a longer period of time. (We retain, however, the liability clearing constraint $L_T = 0$.) We plot the optimal total gross liability and the total squared gross payment for various values of $\lambda$ in Fig. 3. (We do not compare to the pro-rata baseline because it does not seek to make payments small.)

## 6.2 Liability reduction

Suppose that some entities have negative initial net worth. This means that we will not be able to clear all of the liabilities; our goal is then to reduce the liabilities as much as possible, subject to the pro-rata constraint (9), $\mathbf{diag}(L^{\text{init}}\mathbf{1})L_T = \mathbf{diag}(L_T\mathbf{1})L^{\text{init}}$. We consider the same liability matrix as Sect. 6.1, but change the initial cash to

$$c_1 = \max(L^{\text{init}}\mathbf{1} - (L^{\text{init}})^T\mathbf{1} + z, 0), \quad z_i \sim \mathcal{U}(-5, 5), \quad i = 1, \dots, n, \tag{18}$$

where $\mathcal{U}(-5, 5)$ is the uniform distribution on $[-5, 5]$, which in our case leads to 49 entities with negative net worth. We consider the stage costs

$$g_t(c_t, L_t, P_t) = \begin{cases} \mathbf{1}^T L_t \mathbf{1} & P_t\mathbf{1} \le c_t/2, \\ +\infty & \text{otherwise,} \end{cases} \qquad g_T(c_T, L_T) = \mathbf{1}^T L_T \mathbf{1}.$$

The stage cost is the total gross liability, plus the indicator function of the constraint that each entity pays out no more than half of its available cash in each time period. We adjust the pro-rata baseline to

$$P_t = \min(\mathbf{diag}(c_t/2)\Pi, L_t),$$

so that each entity pays no more than half its available cash, and increase the time horizon to $T = 20$. The results are displayed in Fig. 4. The optimal scheme is able to

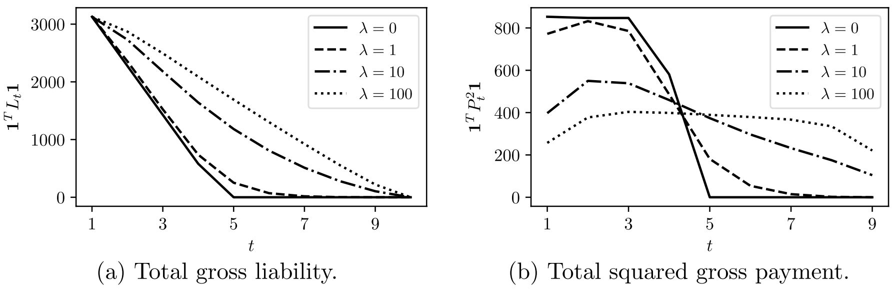


**Fig. 3** Minimizing the sum of total gross liability plus total squared gross payment for various values of $\lambda$

reduce the liabilities faster than the baseline; both methods clear all but around 350 of the original 2000 liabilities. (In Sect. 7.5 we will see an extension that directly includes the number of non-cleared liabilities in the stage cost.)

### 6.3 Exogenous unknown inputs

Next we consider the case where there are exogenous unknown inputs to the dynamics. The cash flows and change in liabilities are sampled according to

$$w_t = z_1, \quad (W_t)_{ij} = \begin{cases} z_2/10 & (L^{\text{init}})_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad t = 1, \dots, T-1,$$

where $\log(z_1) \sim \mathcal{N}(0, I)$ and $\log(z_2) \sim \mathcal{N}(0, 1)$. At each time step $t$, we use the mean of the future inputs as the forecast, or

$$\hat{w}_{\tau|t} = e^{1/2}\mathbf{1}, \quad \hat{W}_{\tau|t} = \begin{cases} e^{1/2}/10 & (L^{\text{init}})_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad \tau = t+1, \dots, T-1.$$

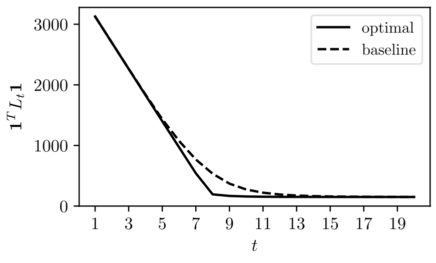
We sample the initial cash vector according to

$$c_1 = \max(L^{\text{init}}\mathbf{1} - (L^{\text{init}})^T\mathbf{1} + z, 0), \quad z_i \sim \mathcal{U}(-5, 0), \quad i = 1, \dots, n.$$

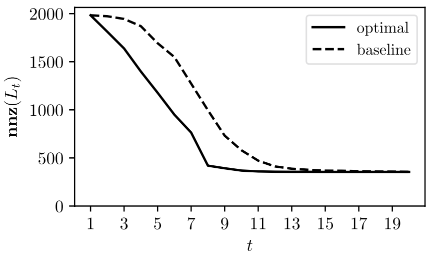
We use the stage cost function $g_t(c_t, L_t, P_t) = \mathbf{1}^T L_t \mathbf{1}$ and the MPC policy described in Sect. 5.2. We compared the shrinking horizon MPC policy with the (modified) pro-rata baseline policy described in Sect. 5.2. The results are displayed in Fig. 5; note that the total gross liability appears to reach a statistical steady state and the liabilities can never be fully cleared. The MPC policy appears to be better than the baseline at reducing liabilities.

## 7 Extensions and variations

In this section we mention some extensions and variations on the formulations described above.



(a) Total gross liability.



(b) Number of non-cleared liabilities.

**Fig. 4** Liability reduction example

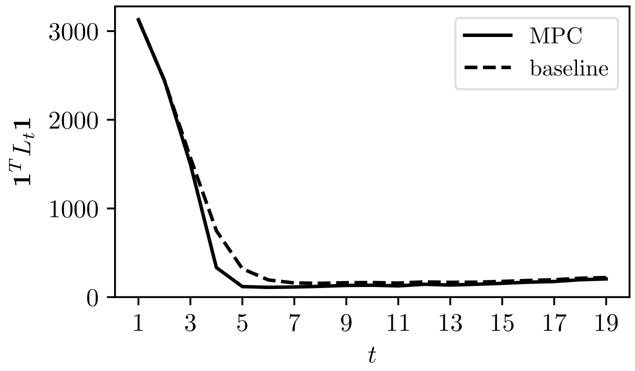


**Fig. 5** Liability control with exogenous inputs

### 7.1 Bailouts

We can add an additional term to the dynamics that injects cash into the entities at various times, with presumably very high cost in the objective. With a linear objective term with sufficiently high weight, the bailout cash injections are zero, if it is possible to clear the liabilities without cash injection. We note that bailouts have been considered in Capponi and Chen (2015, Sect. 2.2).

### 7.2 Minimum time to clear liabilities

Instead of the time-separable cost function given in (8), we take as the objective the number of steps needed to clear all liabilities. That is, our objective is $T^{\mathrm{clr}}$, defined as the minimum value of $t$ for which $L_t = 0$ is feasible. It is easily shown that $T^{\mathrm{clr}}$ is a quasi-convex function of the liability sequence $L_1, \dots, L_T$ (Boyd and Vandenberghe 2004, Sect. 4.2.5), so this problem is readily solved using bisection, solving no more than $\log_2 T$ convex problems. If the liabilities cannot be cleared in up to $T$ steps then we can find a $T$ such that they can be cleared using the techniques described in Agrawal and Boyd (2020, Sect. 3).

### 7.3 Non-time-separable cost

The cost function in our basic formulation (8) is separable, *i.e.*, a sum of terms for each $t = 1, \dots, T$. This can be extended to include non-separable cost functions. We describe a few of these below. They are convex, but non-convex versions of the same objectives can also be employed, at the cost of computational efficiency to solve the problem globally.

**Smooth payments** Adding the term $\sum_{t=2}^{T-1} \|P_t - P_{t-1}\|_F^2$ to the cost, where $\|A\|_F$ is the Frobenius norm, *i.e.*, the square root of the sum of the squared entries of $A$, causes smooth transitions in the payment matrix. This cost is sometimes called the Dirichlet energy (Boyd and Vandenberghe 2018, Sect. 7.3) or a Laplacian regularization term (Biemond et al. 1990).

**Piecewise constant payments** Adding the term $\sum_{t=2}^{T-1} \|P_t - P_{t-1}\|_1$, where $\|A\|_1$ is the sum of absolute values of the entries of $A$, to the cost encourages the payment matrix to change in as few entries as possible between time steps. This cost is sometimes called the total variation penalty (Rudin et al. 1992).

**Global payment restructuring** Adding the term $\sum_{t=2}^{T-1} \|P_t - P_{t-1}\|_F$ to the cost encourages the entire payment matrix to change at as few time steps as possible (Danaher et al. 2014).

**Per-entity payment restructuring** Adding the term $\sum_{t=2}^{T-1} \sum_{i=1}^{n} \|P_t^T e_i - P_{t-1}^T e_i\|_2$, where $e_i$ is the $i$th unit vector, to the cost encourages the rows of the payment matrix, *i.e.*, the payments made by each entity, to change at as few time steps as possible. This penalty is sometimes called a group lasso penalty (Yuan and Lin 2006).

## 7.4 Infinite time liability control

In Sect. 5.2 we described what is often called shrinking horizon control, because at time period $t$, we solve for a sequence of payments $P_t, \dots, P_{T-1}$ over the remaining horizon; the number of payments we optimize over (*i.e.*, $T - t$) shrinks as $t$ increases. This formulation assumes there is a fixed horizon $T$.

It is also possible to consider a formulation with no fixed horizon $T$; the liability clearing is done over periods $t = 1, 2, \dots$ without end. The exogenous inputs $w_t$ and $W_t$ also continue without end. Since we have exogenous inputs, we will generally not be able to clear the liabilities; our goal is only to keep the liabilities small, while making if possible small payments. In this case we have a traditional infinite horizon control or regulator problem. In economics terms, this is an equilibrium payment scheme.

The MPC formulation is readily extended to this case, and is sometimes called receding horizon control (RHC), since we are always planning out $T$ steps from the current time $t$. It is common to add a clearing constraint at the horizon in infinite time MPC or RHC formulations (Rawlings and Mayne 2009, Sect. 2.2).

## 7.5 Minimizing the number of non-cleared liabilities

Another reasonable objective to consider is the number of non-cleared (*i.e.*, remaining) liabilities. In this case, the only cost is the number of nonzero entries in $L_T$. This problem is non-convex, but it can be readily formulated as a mixed-integer convex program (MICP), and solved, albeit slowly, using standard MICP techniques such as branch-and-bound (Land and Doig 1960). It can also be approximately solved much quicker using heuristics, such as iterative weighted $\ell_1$-minimization (Candes et al. 2008).

As a numerical example, we consider a smaller version of the initial liability matrix used in Sect. 6, with $n = 40$ and 400 nonzero initial liabilities. We sample the initial cash according to (18), so that the liabilities cannot be fully cleared, and use a time horizon $T = 10$. Minimizing the sum of total gross liabilities takes 0.05 seconds, resulting in 46 non-cleared liabilities and a final total gross liability of 22.52. By contrast, minimizing the number of non-cleared liabilities takes 22.93 seconds, resulting in only 10 non-cleared liabilities and a final total gross liability of 29.78. (The increase in computation time of a mixed-integer convex optimal control problem, compared to a convex optimal control problem of the same size, increases rapidly with problem size.)

As an extension of minimizing the number of non-cleared liabilities, we can consider minimizing the number of non-cleared entities. If the $i$th row of $L_t$ is zero, it means that entity $i$ does not owe anything to the others, and we say this entity is cleared. We can easily add the number of non-cleared entities to our stage cost, using a mixed-integer convex formulation.

## 7.6 Distributed algorithm

As stated, the liability control problem (8) requires global coordination, *i.e.*, full knowledge of the cash held and the liabilities between the entities throughout the optimization procedure. In many settings where cash, liabilities, or payments cannot be publicly disclosed, this is not possible.

It is possible to solve the liability control problem in a distributed manner where each entity only knows its cash and the payments and liabilities it is involved in during the optimization procedure. That is, entity $i$ only needs to know $(c_t)_i$, the $i$th row and column of $L_t$, and the $i$th row and column of $P_t$.

We can do this by adding a variable $\tilde{P}_t \in \mathbf{R}_+^{n \times n}$, the constraint $\tilde{P}_t = P_t^T$, $t = 1, \dots, T$, and replacing the cash dynamics (3) with

$$c_{t+1} = c_t - P_t\mathbf{1} + \tilde{P}_t\mathbf{1}, \quad t = 1, \dots, T-1.$$

(We can think of $\tilde{P}_t^T$ as a copy of $P_t$, and the constraint $\tilde{P}_t = P_t^T$ as a consensus constraint, *i.e.*, that the two variables must have the same value.) Then, by applying the alternating direction method of multipliers (ADMM) to the splitting $(c_t, L_t, P_t)$ and $\tilde{P}_t$, we arrive at a distributed algorithm for the problem (Boyd et al. 2011). Each iteration of the algorithm involves three steps; 1) each entity solves a separate control problem to compute their cash, outbound liabilities, and outbound payments; 2) each entity solves a separate least squares problem that depends on their inbound payments; and 3) each entity performs a separate dual variable update. When the stage cost is convex, this algorithm is guaranteed to converge to a (global) solution (Boyd et al. 2011,Appendix A). Each step of the algorithm only requires coordination between entities connected in the liability graph, and hence preserves some level of privacy. Similar ideas have been used to develop distributed privacy-preserving implementations of predictive patient models across hospitals (Jochems et al. 2016) and energy management across microgrid systems (Liu et al. 2017).

## 8 Conclusions

We have formulated the multi-period liability clearing problem as a convex optimal control problem. This formulation has many advantages, such as handling many constraints and objectives, low computational complexity, and the ability to handle liabilities (or other quantities) that change over time via model predictive control. While the method requires that the stage cost be a convex function, we have described many practical examples where this holds.

## A Sparsity preserving formulation

In this section we describe a sparsity-preserving formulation of problem (8). We make use of the fact that $L_t$ and $P_t$ are at least as sparse as $L^{\text{init}}$ (see Sect. 3).

First, let $m = \mathbf{nnz}(L^{\text{init}})$ and $I_k \in \{1, \dots, n\} \times \{1, \dots, n\}$, $k = 1, \dots, m$, be the sparsity pattern of $L^{\text{init}}$, meaning $(L^{\text{init}})_{ij} = 0$ for all $(i,j) \notin I_k$, $k = 1, \dots, m$. Instead of working with the matrix variables $L_t$ and $P_t$, we work with the vector variables $l_t \in \mathbf{R}^m_+$ and $p_t \in \mathbf{R}^m_+$, which represent the nonzero entries of $L_t$ and $P_t$ (in the same order). That is,

$$(l_t)_k = (L_t)_{ij}, \quad (p_t)_k = (P_t)_{ij}, \quad (i,j) = I_k, \quad k = 1, \dots, m.$$

The initial liability is given by $l^{\text{init}} \in \mathbf{R}^m_+$, which contains the nonzero entries of $L^{\text{init}}$. The sparsity preserving formulation of the optimal control problem (8) has the form

$$\begin{array}{ll} \text{minimize} & \sum_{t=1}^{T-1} g_t(c_t, l_t, p_t) + g_T(c_T, l_T) \\ \text{subject to} & l_{t+1} = l_t - p_t, \quad t = 1, \dots, T-1, \\ & c_{t+1} = c_t - S^{\text{row}} p_t + S^{\text{col}} p_t, \quad t = 1, \dots, T-1, \\ & S^{\text{row}} p_t \le c_t, \quad t = 1, \dots, T-1, \\ & p_t \ge 0, \quad t = 1, \dots, T-1, \\ & l_t \ge 0, \quad t = 1, \dots, T, \\ & c_1 = c^{\text{init}}, \quad l_1 = l^{\text{init}}, \quad c_T \ge 0, \end{array} \tag{19}$$

where $S^{\text{row}} \in \mathbf{R}^{n \times m}$ sums the rows of $P_t$, *i.e.*, $S^{\text{row}} p_t = P_t \mathbf{1}$, and $S^{\text{col}} \in \mathbf{R}^{n \times m}$ sums the columns of $P_t$, *i.e.*, $S^{\text{col}} p_t = P_t^T \mathbf{1}$. The cost functions are applied only to the nonzero entries of $L_t$ and $P_t$, so they take the form $g_t : \mathbf{R}^n_+ \times \mathbf{R}^m_+ \times \mathbf{R}^m_+ \to \mathbf{R} \cup \{+\infty\}$ and $g_T : \mathbf{R}^n_+ \times \mathbf{R}^m_+ \to \mathbf{R} \cup \{+\infty\}$. Problem (19) has just $2T(n+m)$ variables, which can be much fewer than the original $2T(n+n^2)$ variables when $m \ll n^2$.

**Acknowledgements** The authors would like to thank Zachary Feinstein and Luitgard Veraart for their helpful discussion and comments on an early draft of the paper. The authors would also like to thank Daniel Saedi for general discussions about banking. Shane Barratt is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

## References

Agrawal A, Boyd S (2020) Disciplined quasiconvex programming. Optim Lett 14:1643–1657

Agrawal A, Verschueren R, Diamond S, Boyd S (2018) A rewriting system for convex optimization problems. J Control Decis 5(1):42–60

Agrawal A, Amos B, Barratt S, Boyd S, Diamond S, Kolter JZ (2019) Differentiable convex optimization layers. In: Advances in neural information processing systems, pp 9558–9570

Allen F, Gale D (2000) Financial contagion. J Polit Econ 108(1):1–33

Amini H, Feinstein Z (2020) Optimal network compression. https://arxiv.org/abs/2008.08733, August

Banerjee T, Feinstein Z (2019) Impact of contingent payments on systemic risk in financial networks. Math Financ Econ 13(4):617–636

Banerjee T, Bernstein A, Feinstein Z (2018) Dynamic clearing and contagion in financial networks. arXiv preprint arXiv:1801.02091

Bardoscia M, Ferrara G, Vause N, Yoganayagam M (2019) Full payment algorithm. Available at SSRN

Bekaert G, Hodrick R (2012) International financial management, 2nd edn. Pearson Education, London

Bemporad A (2006) Model predictive control design: New trends and tools. In: IEEE conference on decision and control, pp 6678–6683

Biemond J, Lagendijk R, Mersereau R (1990) Iterative methods for image deblurring. Proc IEEE 78(5):856–883

Blackmore L, Açikmeşe B, Scharf D (2010) Minimum-landing-error powered-descent guidance for Mars landing using convex optimization. J Guid Control Dyn 33(4):1161–1171

Board of Governors of the Federal Reserve System. Reserve requirements. https://www.federalreserve.gov/monetarypolicy/reservereq.htm, March 2020

Boss M, Elsinger H, Summer M, Thurner S (2004) Network topology of the interbank market. Quant Financ 4(6):677–684

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Boyd S, Vandenberghe L (2018) Introduction to applied linear algebra: vectors, matrices, and least squares. Cambridge University Press, Cambridge

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends® Mach Learn 3(1):1–122

Boyd S, Busseti E, Diamond S, Kahn R, Koh K, Nystrup P, Speth J (2017) Multi-period trading via convex optimization. Found Trends® Optim 3(1):1–76

Brouwer D (2009) System and method of implementing massive early terminations of long term financial contracts. US Patent 7,613,649

Candes E, Wakin M, Boyd S (2008) Enhancing sparsity by reweighted $\ell_1$ minimization. J Fourier Anal Appl 14(5–6):877–905

Capponi A, Chen P-C (2015) Systemic risk mitigation in financial networks. J Econ Dyn Control 58:152–166

Cho E, Thoney K, Hodgson T, King R (2003) Supply chain planning: Rolling horizon scheduling of multi-factory supply chains. In: Proceedings of the conference on winter simulation: driving innovation, pp 1409–1416

Cifuentes R, Ferrucci G, Shin HS (2005) Liquidity risk and contagion. J Eur Econ Assoc 3(2–3):556–566

Danaher P, Wang P, Witten D (2014) The joint graphical lasso for inverse covariance estimation across multiple classes. J R Stat Soc: Ser B (Stat Methodol) 76(2):373–397

Diamond S, Boyd S (2016) CVXPY: a Python-embedded modeling language for convex optimization. J Mach Learn Res 17(83):1–5

Diamond D, Dybvig P (1983) Bank runs, deposit insurance, and liquidity. J Polit Econ 91(3):401–419

D'Errico M, Roukny T (2017) Compressing over-the-counter markets. Technical report, European Systemic Risk Board

Eisenberg L, Noe T (2001) Systemic risk in financial systems. Manag Sci 47(2):236–249

Elsinger H (2009) Financial networks, cross holdings, and limited liability. Working Papers, Oesterreichische Nationalbank (Austrian Central Bank) (156)

Falcone P, Borrelli F, Asgari J, Tseng H, Hrovat D (2007) Predictive active steering control for autonomous vehicle systems. IEEE Trans Control Syst Technol 15(3):566–580

Federal Deposit Insurance Corporation. Statistics at a glance. https://www.fdic.gov/bank/statistical/stats/2019dec/industry.pdf, December 2019

Feinstein Z (2019) Obligations with physical delivery in a multilayered financial network. SIAM J Financ Math 10(4):877–906

Feinstein Z, Rudloff B, Weber S (2017) Measures of systemic risk. SIAM J Financ Math 8(1):672–708

Feinstein Z, Pang W, Rudloff B, Schaanning E, Sturm S, Wildman M (2018) Sensitivity of the Eisenberg-Noe clearing vector to individual interbank liabilities. SIAM J Financ Math 9(4):1286–1325

Fu A, Narasimhan B, Boyd S (2019) CVXR: an R package for disciplined convex optimization. J Stat Softw

Glasserman P, Young H (2015) How likely is contagion in financial networks? J Bank Financ 50:383–399

Grant M, Boyd S (2008) Graph implementations for nonsmooth convex programs. In: Recent advances in learning and control. Lecture Notes in Control and Information Sciences. Springer, Berlin, pp 95–110

Grant M, Boyd S (2014) CVX: Matlab software for disciplined convex programming, version 2.1

Harris T, Ross F (1955) Fundamentals of a method for evaluating rail net capacities. Technical report, RAND Corp, Santa Monica, CA

Jochems A, Deist TM, Van Soest J, Eble M, Bulens P, Coucke P, Dries W, Lambin P, Dekker A (2016) Distributed learning: developing a predictive model based on data from multiple hospitals without data leaving the hospital–a real life proof of concept. Radiother Oncol 121(3):459–467

Kahn A (1962) Topological sorting of large networks. Commun ACM 5(11):558–562

Khabazian A, Peng J (2019) Vulnerability analysis of the financial network. Manag Sci 65(7):3302–3321

Kusnetsov M, Veraart L (2019) Interbank clearing in financial networks with multiple maturities. SIAM J Financ Math 10(1):37–67

Land A, Doig A (1960) An automatic method of solving discrete programming problems. Econometrica 28(3):497–520

Liu Y, Gooi HB, Xin H (2017) Distributed energy management for the multi-microgrid system based on ADMM. In: Power & energy society general meeting. IEEE, pp 1–5

Lofberg J (2004) YALMIP: a toolbox for modeling and optimization in MATLAB. In: IEEE international conference on robotics and automation. IEEE, pp 284–289

Ma Y, Borrelli F, Hencey B, Coffey B, Bengea S, Haves P (2011) Model predictive control for the operation of building cooling systems. IEEE Trans Control Syst Technol 20(3):796–803

Mattingley J, Boyd S (2009) Automatic code generation for real-time convex optimization. In: Convex optimization in signal processing and communications, pp 1–41

Mattingley J, Boyd S (2012) CVXGEN: a code generator for embedded convex optimization. Optim Eng 13(1):1–27

Mattingley J, Wang Y, Boyd S (2011) Receding horizon control. IEEE Control Syst Mag 31(3):52–65

Moehle N, Busseti E, Boyd S, Wytock M (2019) Dynamic energy management. In: Large scale optimization in supply chains and smart manufacturing. Springer, Berlin, pp 69–126

MOSEK optimization suite (2020). https://www.mosek.com

O'Kane D (2014) Optimizing the compression cycle: algorithms for multilateral netting in OTC derivatives markets. Available at SSRN 2273802

O'Kane D (2017) Optimising the multilateral netting of fungible OTC derivatives. Quant Financ 17(10):1523–1534

Rawlings J, Mayne D (2009) Model predictive control: theory and design. Nob Hill Publishing, San Francisco

Rogers L, Veraart L (2013) Failure and rescue in an interbank network. Manag Sci 59(4):882–898

Rudin L, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. Phys D: Nonlinear Phenom 60(1–4):259–268

Schuldenzucker S, Seuken S (2019) Portfolio compression in financial networks: incentives and systemic risk. Available at SSRN

Shapiro A (1978) Payments netting in international cash management. J Int Bus Stud 9(2):51–58

Soltani M, Wisniewski R, Brath P, Boyd S (2011) Load reduction of wind turbines using receding horizon control. In: *IEEE international conference on control applications*. IEEE, pp 852–857

Udell M, Mohan K, Zeng D, Hong J, Diamond S, Boyd S (2014) Convex optimization in Julia. Workshop on high performance technical computing in dynamic languages

Veraart L (2019) When does portfolio compression reduce systemic risk? *Available at SSRN 3488398*

Wang Y, Boyd S (2009) Performance bounds for linear stochastic control. Syst Control Lett 58(3):178–182

Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. J R Stat Soc: Ser B (Stat Methodol) 68(1):49–67