A front fixing approach to solving sovereign default models

Stephan Hobler

30th October 2025

Abstract

This paper proposes a robust and efficient method of solving models with endogenous borrowing constraints and strategic default in continuous-time. Using a change of variable the free-boundary problem is transformed into a boundary value problem on a fixed domain in which default thresholds enter explicitly as endogenous variables. Default thresholds can then be solved using algorithms that are analogous to solving for the fixed point of prices in standard general equilibrium models. I demonstrate the approach using two sovereign default models featuring short-term and long-term debt, respectively. At last, I show how the method can be applied to time-dependent problems featuring transition dynamics.

1 Introduction

Strategic default models in continuous-time present two distinct computational challenges. First, the practitioner must jointly solve a coupled system of partial differential equations (PDEs) together with the endogenous default decisions, the so-called *free boundary*. Second, borrowing constraints are endogenous and themselves a function of the default decision. This paper proposes a method for solving this class of free boundary problems with endogenous borrowing constraints that is robust, efficient and generalizable to other settings. The idea is to transform the *free boundary* problem into a *fixed boundary* problem in which the boundary enters the PDE as a parameter. Importantly, this fixes the computational domain and makes the problem amenable to standard finite difference methods that can be applied with minimal changes.

The intuition is analogous to the analytical approach to solving optimal stopping time problems. Default decisions can be characterized by the distance of current asset holdings to a default threshold. It is convenient to define a new state variable in terms of the distance to the default threshold. In this transformed system, the default threshold is no longer free but is at zero by definition. That is, in the transformed system the location of the boundary is known ex-ante and for a given guess of the default threshold the computational domain is fixed. Standard finite difference methods that are consistent, monotone and stable in the sense of Barles & Souganidis (1991) can be applied without any modification to solve the resulting PDEs for a given guess of the default threshold. Finally, the default threshold is updated using standard Newton methods to ensure that the boundary conditions derived from the optimal stopping time problem are satisfied.

The resulting structure will be familiar to economists and is analogous to solving for prices in general equilibrium models. In the transformed system, default thresholds enter as parameters in the value function which need to satisfy certain "equilibrium conditions" such as value matching. Conceptually, this is equivalent to the way prices enter the value function as parameters and are pinned down by a set of general equilibrium conditions to equilibrate demand and supply. Standard root finding algorithms such as Newton methods can be applied.

I benchmark my method against Bornstein (2020), who develops an algorithm to solve a continuous-time version of Arellano (2008). Bornstein (2020) proposes a *trial* method in which he guesses the default threshold as a function of income and updates the threshold based on the relative value of being solvent or in default. Relative to this paper I see my contribution as follows. First, the front fixing approach keeps the computational domain fixed which simplifies the computation of the inner loop. Second, I do not restrict the borrowing constraints to lie on a pre-specified grid. This increases the accuracy of the solution. What is more, as models can oscillate because of the discretisation of the grid (e.g. see Bornstein (2020)) this helps with convergence in models with long-run debt. Third, by transforming the problem into one that is continuous in the default threshold I make the problem amenable to gradient based methods which are efficient and provide a well-informed updating rule. In contrast, the updating rule by Bornstein (2020) potentially throws away a lot of useful information and might achieve less robust convergence in more complex settings. By casting the problem as a system of smooth non-linear equations the researcher can further fall back on powerful, well-established solvers to find the solution to the problem and resort to a vast literature on ensuring robust convergence.

Separate from solving the free boundary problem, I find that to robustly solve sovereign default models with long-term debt it is necessary to treat the non-linear term in the bond price equation fully implicitly.[1] It is important to jointly solve the coupled PDEs of the value function and the bond pricing equation and use the full, though highly sparse, Jacobian for updating. Further, to ensure robustness I use a continuation method. To still obtain efficient computation I use sparse automatic differentiation tools to compute the Jacobian and treat the jump terms explicitly reduce the computational cost of each linear solve.

Relation to other approaches. The additional difficulty of strategic default models with endogenous borrowing constraints lies in the fact that borrowing constraints restrict the computational domain of the problem. That is, in addition to the standard conditions for optimal stopping time problems such as value matching we have an additional state boundary constraint that dpends on the free boundary. To the best of my knowledge, other methods such as LCP reformulations or Operator Splitting methods cannot straightforwardly deal with this. The reason is that one needs to impose the boundary condition at the free boundary. Fixing the free boundary ex-ante simplifies this substantially. A potential alternative are penalty methods as applied in Hurtado et al. (2022). However, here the difficulty lies in choosing a common borrowing limit that is feasible for all income levels.

Literature. The front-fixing approach is not new, though its application to sovereign default models with endogenous borrowing constraints to the best of my knowledge is. Landau (1950) first proposed the method in thermodynamics and Crank (1957) first applied it to finite difference methods. The front-fixing method has been introduced to option pricing by Wu & Kwok (1997) and since been successfully applied to various option pricing problems by Nielsen et al. (2001), Zhu et al. (2003), Duffy (2006), Heidari & Azari (2017), Fazio et al. (2021), Company et al. (2021). Closest to this paper is Bornstein (2020). I propose the front-fixing framework as an alternative to his method.

[1] Gomez (2024) makes a related point in a macro-finance application.

2 Application 1: Sovereign default with short-term debt

2.1 Economic model

The first application is a sovereign default model with short-term debt. I consider a continuous-time version of Arellano (2008) which is identical to Bornstein (2020). For a detailed exposition I refer the reader to Bornstein (2020). At each instant a sovereign decides whether to repay their instantly maturing debt or whether to default. The relevant state variables are assets a (i.e. savings or negative debt) and income y. The decisions are over consumption $c \geq 0$ and a default decision $D \in \{0,1\}$. Default can be characterised by a threshold rule $\underline{a}(y)$ such that $D(a,y) = 0$ for all $a \geq \underline{a}(y)$ and $D(a,y) = 1$ for all $a < \underline{a}(y)$. Income is subject to a compound Poisson process with arrival rate λ_y and a stochastic jump size with density $f(y,y')$. During default a sovereign loses access to financial markets, incurs a default cost $\phi(y)$ and regains access to the financial markets at a rate λ_D at which point it becomes a zero asset sovereign. A competitive financial sector determines the short term interest rate $r(a,y)$. The set-up yields two value functions, v and w, the value of being solvent and the value of being in default, respectively.

Concisely, the system of HJBs for the value of being solvent v and of being in default w is defined by

$$\begin{aligned}\rho v(a,y) = \max\Big\{\rho w(y),\ \max_c u(c) + \partial_a v(a,y)\left[y - c + r(a,y)a\right] \\ + \lambda_y \int_0^\infty \left[v(a,y') - v(a,y)\right] f(y,y')dy'\Big\}\end{aligned} \tag{2.1}$$

$$\rho w(y) = u(y - \phi(y)) + \lambda_y \int_0^\infty \left[w(y') - w(y)\right] f(y,y')dy' + \lambda_D\left[v(0,y) - w(y)\right] \tag{2.2}$$

$$v(\underline{a}(y), y) = w(y) \tag{2.3}$$

subject to the income-specific borrowing (state) constraint

$$a \geq \underline{a}(y) \quad \forall y \in \mathbb{R}^+,$$

and the endogenous interest rate schedule reflecting default

$$r(a,y) = r_f + \lambda_y \int_0^{+\infty} D(a,y') f(y,y')dy', \tag{2.4}$$

and where (2.3) denotes the so-called value matching condition defining the location of the default

threshold, i.e. the *free boundary*. The default threshold is implicitly defined by (2.3) and coincides with the borrowing limit as creditors will not give out loans to an instantly defaulting sovereign.

Three components make this problem non-standard: (i) the borrowing limit varies by income state (ii) the borrowing limit $\underline{a}(y)$ must be consistent with the default decision $D(a,y)$ and (iii) the interest rate schedule $r(a,y)$ reflects the sovereign's likelihood of default. The value matching condition (2.3) makes it clear why this is non-trivial to solve. We need v and w to compute $\underline{a}(y)$ but we need $\underline{a}(y)$ to compute v and w.

A full statement of the problem would also include a set of boundary conditions. For conciseness, let us focus on the state boundary constraint

$$\partial_a v(\underline{a}(y),y) \geq u'(y + r(a,y)\underline{a}(y)), \quad \text{for all } y \in \mathbb{R}^+. \tag{2.5}$$

This highlights the complication with other existing methods such as the Lincear complementarity Problem (LCP) solver. The boundary condition explicitly depends on the free boundary $\underline{a}(y)$. The explicit dependence on the boundary condition thus necessitates to keep track of the free boundary unlike a LCP solver.[2]

2.2 Front fixing

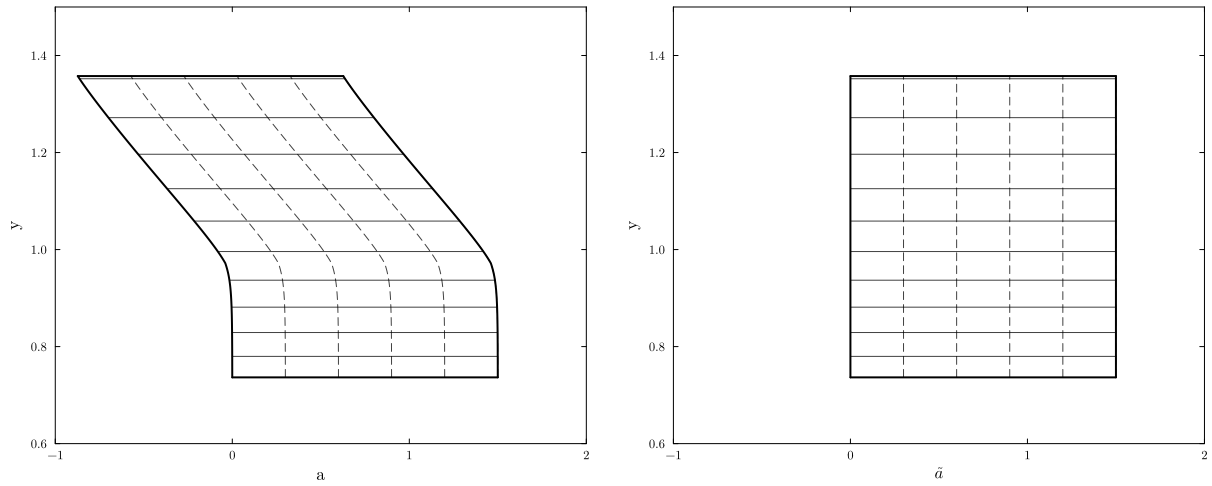
The fundamental idea is to convert the *free* boundary problem into a problem on a *fixed* domain. Specifically, we can perform a change of variable to obtain a fixed boundary problem where default occurs at a pre-defined value. The default threshold then shows up explicitly inside the HJB equations as parameters and not as a free boundary condition that needs to be solved for. There is a close parallel to the analytical solution to optimal stopping time problems. Appendix A expands on this point more formally using a simple option problem as an example. This makes the problem simpler in one dimension and more complex in another as we now need to solve for endogenous variables.

[2] A linear complementarity problem (LCP) is given by the following conditions

$$\begin{aligned} z'(Bz+q) &= 0 \\ z &\geq 0 \\ Bz+q &\geq 0. \end{aligned}$$

However, in sovereign default models the matrix B is itself a function of the default threshold, i.e. whether $z \lesseqgtr 0$. This is no longer a linear problem.

Change of variables. Define the transformed state variable $\tilde{a} := a - \underline{a}(y)$ and a new function $\varphi\left(a - \underline{a}(y), y; \underline{\boldsymbol{a}}\right) \equiv v(a, y)$ where we will treat the default threshold $\underline{\boldsymbol{a}}$ explicitly as a (infinite-dimensional) parameter.[3] Crucially, this change of variable *fixes* the default threshold at $\tilde{a} = a - \underline{a}(y) = 0$ and the sovereign defaults if $\tilde{a} < 0$. In this sense, the boundary is known in the transformed system to be at $\tilde{a} = 0$ and the computational domain is fixed at $\tilde{a} \geq 0$. We can now solve the system of PDEs as per usual for a given $\underline{a}(y)$ using finite differences Achdou et al. (2021).[4]



(a) Untransformed grid.

(b) Transformed grid.

Figure 1: Illustration of grid transformation.

Plugging our transformation into the HJB we get for $\tilde{a} \geq 0$

$$\begin{aligned}\rho\varphi\left(\tilde{a}, y\right) = \max_c u(c) + \partial_{\tilde{a}}\varphi\left(\tilde{a}, y\right)\left[y + r\left(\tilde{a}, y\right)\left(\tilde{a} + \underline{a}(y)\right) - c\right] \\ + \lambda_y \int_0^{+\infty} \left[\varphi\left(\tilde{a} + \underline{a}(y) - \underline{a}(y'), y'\right) - \varphi\left(\tilde{a}, y\right)\right] f(y, y')\, dy'\end{aligned} \tag{2.6}$$

$$\rho w(y) = u(y - \phi(y)) + \lambda_y \int_0^{\infty} \left[w(y') - w(y)\right] f(y, y')\, dy' + \lambda_D \left[\varphi(-\underline{a}(y), y) - w(y)\right] \tag{2.7}$$

$$\varphi(0, y) = w(y) \tag{2.8}$$

subject to $\tilde{a} \geq 0$. The interest rate schedule is given by

$$r(\tilde{a}, y) = r_f + \lambda_y \int_{y' \in \mathbb{R}^+} \mathbf{1}\{\tilde{a} + \underline{a}(y) - \underline{a}(y')\} f(y, y')\, dy'. \tag{2.9}$$

[3] Other change of variables are equally valid. E.g. we may define $\tilde{a} = \frac{a - \underline{a}(y)}{\overline{a} - \underline{a}(y)}$. This did not change the main results for a sufficiently fine grid.

[4] If the model featured other boundary conditions, e.g. smooth pasting, these could be explicitly imposed at the fixed boundary $\tilde{a} = 0$ at no additional difficulty. We thus see that the approach is easily extended to richer environments.

Derivations are delegated to Appendix B.

Algorithm: A solution to the system of equations is one that is consistent with the HJB equations (2.6)-(2.7), the interest rate schedule (2.9) and satisfies the value matching condition (2.8). This suggests the following algorithm.

Discretize grids a and y as well as the stochastic process over y'. Initialize an arbitrary guess for $\varphi(\tilde{a}, y)$ and $w(y)$.

while $||\varphi(0, y) - w(y)||_\infty > tol$ **do**

1. Given $\underline{a}(y)$ solve for $r(\tilde{a}, y)$ using (2.9).;
2. Update $\varphi(\tilde{a}, y)$ and $w(y)$ using (2.6) and (2.7) by standard methods (see Achdou et al. (2021)).;
3. Check value matching (2.8). If not found root, perform a Newton update on (2.8) and iterate on $\underline{a}(y)$.;

end

Algorithm 1: Solving short-term debt model.

The main benefit of the proposed method is that the borrowing constraint enters as a continuous variable and is not forced to lie on a discretised grid. This improves accuracy of the solution, renders the problem sufficiently smooth and enables the use of powerful, well-establishd non-linear equation solvers to find the solution to (2.8). The resulting PDE is also not substantially more complex to solve (for a given borrowing limit) than the standard model. Further, the fact that $\underline{\boldsymbol{a}}$ enters explicitly as parameters allows for an efficient computation of the Jacobian or Jacobian vector product to apply the Newton or Newton-Krylov step. Jacobians can be computed using repeated application of the change rule and the implicit function theorem. Details are delegated to Appendix C.5.

Numerical implementation. (2.6) is only a slightly modified version of the standard HJB. Specifically, the only changes relative to a model without endogenous default and borrowing constraints are (a) endogenous interest rate schedule $r(\tilde{a}, y)$ (b) a modified jump term including that implies correlated transitions in assets and productivity and (c) the need to interpolate the value function off the grid. While (a) and (c) can be dealt with in a straightforward manner using standard methods (b) necessitates some discussion. Specifically, I define three operators $\mathscr{E}[v](a, y) = \int_0^{+\infty} v(a, y') f(y, y') dy'$,

$\mathscr{M}_1[f](\tilde{a},y) = f(\tilde{a}+\underline{a}(y),y)$, and $\mathscr{M}_2[f](a,y) = f(a-\underline{a}(y),y)$ such that

$$\int_0^{+\infty} \varphi(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y')f(y,y')dy' = (\mathscr{M}_1 \circ \mathscr{E} \circ \mathscr{M}_2)[\varphi](\tilde{a},y). \tag{2.10}$$

We define $\mathscr{M}_1$ and $\mathscr{M}_2$ to be linear interpolation operators such that they correspond to matrices with constant weights. In essence, we map the function into a space in which it is easier to take expectations.[5] That is, the operator $\mathscr{E}$ only requires integration over one and not two dimensions which simplifies the computation. Linear interpolation operators are particularly useful as we can store them as constant basis matrices and pre-compute the matrix corresponding to the operator $\mathscr{M}_1 \circ \mathscr{E} \circ \mathscr{M}_2$. As this is a large and fairly dense matrix this achieves substantial speed gains.

2.3 Comparison with Bornstein (2020)

To benchmark the proposed method I compare the results to the method of Bornstein (2020). I follow his calibration. The results are shown in Figure 2. The two solutions line up closely. However, the front-fixing approach achieves greater accuracy as it does not restrict the borrowing limit to lie on the grid. This is particularly important for the lower productivity grid points.

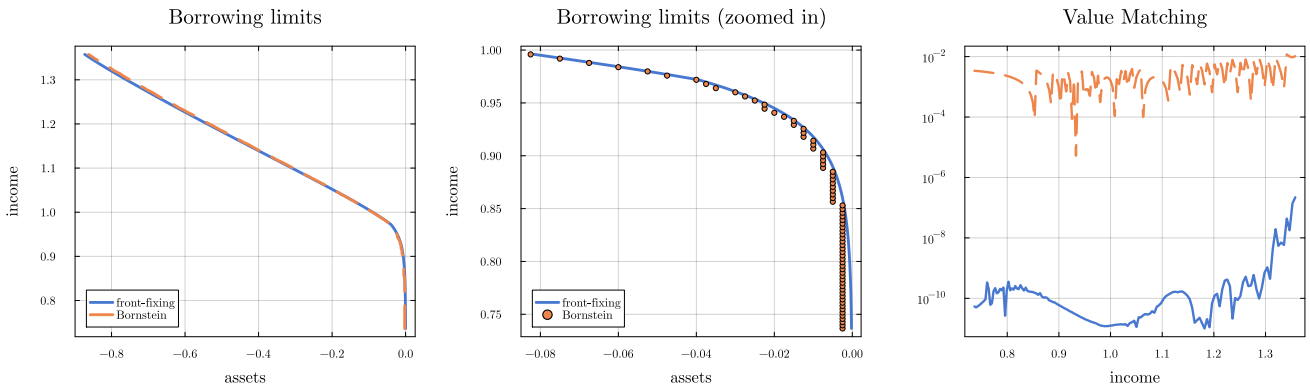


Figure 2: Comparison of *front-fixing* method with *front-tracking* method of Bornstein (2020). *Left:*, *Middle:*, *Right:*

The two methods also produce very similar policy functions as shown in 3. However, the front-fixing method seems to be better able to capture precautionary savings for lower income countries. This comes from the fact that front-fixing does not constrain the borrowing constraint to lie on a grid and the borrowing limit is not constant for lower income levels.

[5] This has some parallels to taking the fast fourier transform to compute expectations.

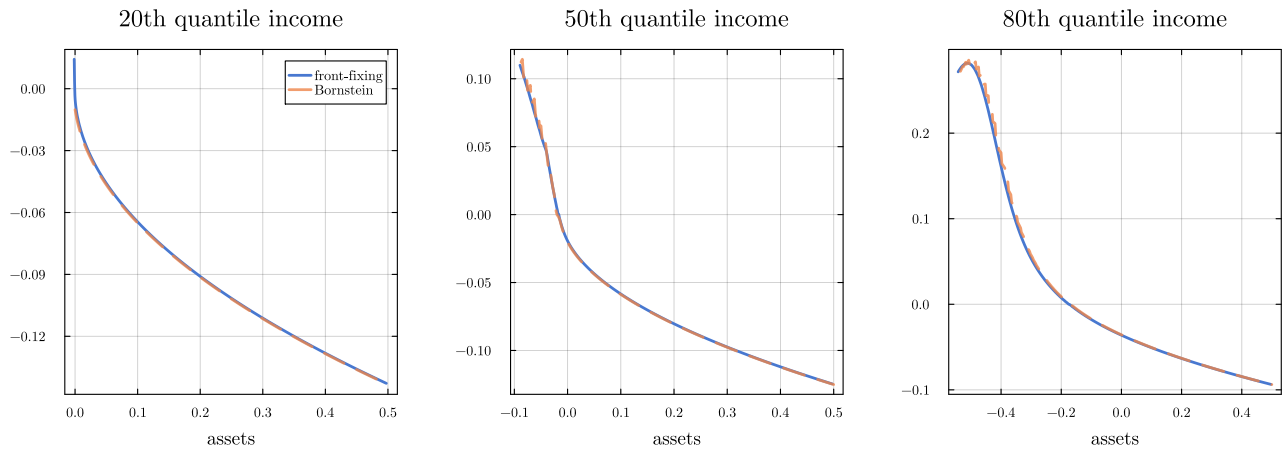


Figure 3: Comparison of *front-fixing* method with *front-tracking* method of Bornstein (2020). Savings policy functions.

3 Sovereign default with long-term debt

Bornstein (2020) discusses two potential sources of non-convergence of the long-term debt sovereign default model: (1) discrete borrowing decision and (2) disrete default frontier. The method of Bornstein (2020) overcomes the convergence issue (1), but not (2). Since the method proposed in this paper does not restrict the default frontier to lie on a pre-specified grid it promises convergence even in models with long-term debt.

3.1 Long-term debt

Instead of instantly maturing debt the sovereign issues bonds at a price $q(a,y)$ with a geometric maturity structure λ_b and coupon payments z. This introduces another forward-looking equation that determines the price of the government bond. The economic model is given by the following coupled PDEs for $v(a,y)$, $w(y)$ and $q(a,y)$

$$\begin{aligned}\rho v(a,y) = \max\Bigg\{\rho w(y),\ \max_c u(c) + \partial_a v(a,y)\left[\frac{y+(z+\lambda_b)a-c}{q(a,y)} - \lambda_b a\right] \\ + \lambda_y \int_0^{+\infty} \left[v(a,y') - v(a,y)\right] f(y,y')dy'\Bigg\}\end{aligned} \tag{3.1}$$

$$\rho w(y) = u(y-\phi(y)) + \lambda_y \int_0^{+\infty} \left[w(y') - w(y)\right] f(y,y')dy' + \lambda_D\left[v(0,y) - w(y)\right] \tag{3.2}$$

$$v(\underline{a}(y),y) = w(y) \tag{3.3}$$

$$r_f q(a,y) = z + \lambda_b(1-q(a,y)) + \partial_a q(a,y)\left[\frac{y+(z+\lambda_b)a - c^*(a,y)}{q(a,y)} - \lambda_b a\right] + \lambda_y \int \left[q(a,y') - q(a,y)\right] f(y,y')dy'. \tag{3.4}$$

We note that equations (3.2) and (3.3) are unchanged. Relative to above, we replace the interest rate schedule with the Feynman-Kac formula for the price of the bond (3.4).

Change of variables. We use the same change of variables as above, i.e. $\tilde{a} = a - \underline{a}(y)$ and define two functions $\varphi(a - \underline{a}(y), y) = v(a,y)$ and $Q(a - \underline{a}(y), y) = q(a,y)$. The respective transformed equations are

$$\rho\varphi(\tilde{a},y) = \max_c u(c) + \partial_{\tilde{a}}\varphi(\tilde{a},y)\left[\frac{y-c}{Q(\tilde{a},y)} + \left(\frac{z+\lambda_b}{Q(\tilde{a},y)} - \lambda_b\right)(\tilde{a} + \underline{a}(y))\right] + \lambda_y \int_0^{+\infty} \left[\varphi\left(\tilde{a} + \underline{a}(y) - \underline{a}(y'), y'\right) - \varphi(\tilde{a},y)\right] f(y,y')dy' \tag{3.5}$$

$$\rho w(y) = u(y - \phi(y)) + \lambda_y \int_0^{+\infty} \left[w(y') - w(y)\right] f(y,y')dy' + \lambda_D\left[\varphi(-\underline{a}(y), y) - w(y)\right] \tag{3.6}$$

$$r_f Q(\tilde{a},y) = z + \lambda_b(1 - Q(\tilde{a},y)) + \partial_{\tilde{a}} Q(\tilde{a},y)\left[\frac{y - c^*(\tilde{a},y)}{Q(\tilde{a},y)} + \left(\frac{z+\lambda_b}{Q(\tilde{a},y)} - \lambda_b\right)(\tilde{a} + \underline{a}(y))\right] + \lambda_y \int_0^{\infty} \left[Q\left(\tilde{a} + \underline{a}(y) - \underline{a}(y'), y'\right) - Q(\tilde{a},y)\right] f(y,y')dy' \tag{3.7}$$

In the default region, the bond price schedule is zero, i.e. $Q(\tilde{a},y) = 0$ for all $\tilde{a} \leq 0$. The value matching condition is

$$\varphi(0,y) = w(y). \tag{3.8}$$

In principle, the procedure is unchanged from above. In practice, long-term bonds introduce additional complexities and require more robust solution techniques. The reason is that the bond pricing equation and the value function equation are highly nonlinear in the bond price $Q(\tilde{a},y)$. Specifically, I employ the following techniques:

1. *Fully implicit:* I treat the coupled system (3.5) and (3.4) jointly as a large system of non-linear equations and apply a Newton method on it to update the guess for $\varphi(\tilde{a},y)$ and $Q(\tilde{a},y)$. To still obtain efficiency, I use an implicit-explicit discretisation method in which I treat jump terms explicitly. I then use sparse automatic differentiation to compute the Jacobian of the system.

2. *Continuation method:* To robustly solve for the bond price I employ adaptive time stepping

techniques similar to Coffey et al. (2003) and Gomez (2024). Intuitively, we are employing a homotopy method of the form $G(x^{(n+1)}, x^{(n)}) = \Delta F(x^{(n+1)}) + (x^{(n+1)} - x^{(n)}) = 0$, where $F(x)$ is the stacked system for φ and Q. For small Δ we are guaranteed to solve the root of G and improve our guess. Continuing along this path of updates proves to be a robust method for solving F.

These numerical intricacies are, however, separate from the front fixing method and also present in Bornstein (2020). In fact, the replication code from Bornstein (2020) also has difficulties to achieve convergence in the bond price schedule at moderate tolerance levels. Appendix C.8 contains further details.

Discretize grids $\tilde{a}$ and y as well as the stochastic process over y'. Initialize a guess for $\varphi(\tilde{a}, y)$, $w(y)$ and $Q(\tilde{a}, y)$. Let $\boldsymbol{x}^{(0)} = [\boldsymbol{\varphi}^{(0)}; \boldsymbol{Q}^{(0)}]$. Set iteration counter to 0.

while $||\varphi(0, y) - w(y)||_\infty > tol$ **do**

1. Use (3.5) and (3.4) to define system $F(\boldsymbol{x}^{(n+1)}, \boldsymbol{x}^{(n)})$ using the Implicit-Explict (IMEX) scheme. Define $G(\boldsymbol{x}^{(n+1)}, \boldsymbol{x}^{(n)}) = F(\boldsymbol{x}^{(n+1)}, \boldsymbol{x}^{(n)}) + \frac{\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}}{\Delta}$;

while $||G|| > tol$ **do**

1.1 Use Newton method (possibly with line search) to solve the root $x^{(n+1)}$ of G.;

1.2 Check whether the Newton method has converged. If not, reduce Δ. If converged, we may increase Δ to achieve faster convergence.;

end

2. Check value matching (3.3). If not found root, perform a Newton update on (3.3) and iterate on $\underline{a}(y)$.;

end

Algorithm 2: Solving long-term debt model.

Enforcing risk-free rate for savers: The above formulation does not capture that the bond schedule is only defined for $a \leq 0$, i.e. for borrrowers. To enforce that I use a penalty method that ensures that the bond price schedule is equal to the risk-free price $\frac{z+\lambda_b}{r_f+\lambda_b}$ for all $a \geq 0$. More generally, one could impose another boundary condition at $\tilde{a} = -\underline{a}(y)$ that ensures that a sovereign will never save – capturing the idea that the bond exclusively captures debt. Alternatively, one could consider alternative change of variables such as $\tilde{a} = \frac{a}{-\underline{a}(y)}$ that would ensure a fixed computational domain on $[-1, 0]$ regardless of the borrowing limit.

3.2 Comparison with Bornstein (2020)

In Figure 4 I compare the solution to the front-fixing method of the long-term debt model to the replication code from Bornstein (2020). The solutions of the respective methods line up pretty well in terms of the borrowing limit while the front-fixing method achieves greater accuracy.

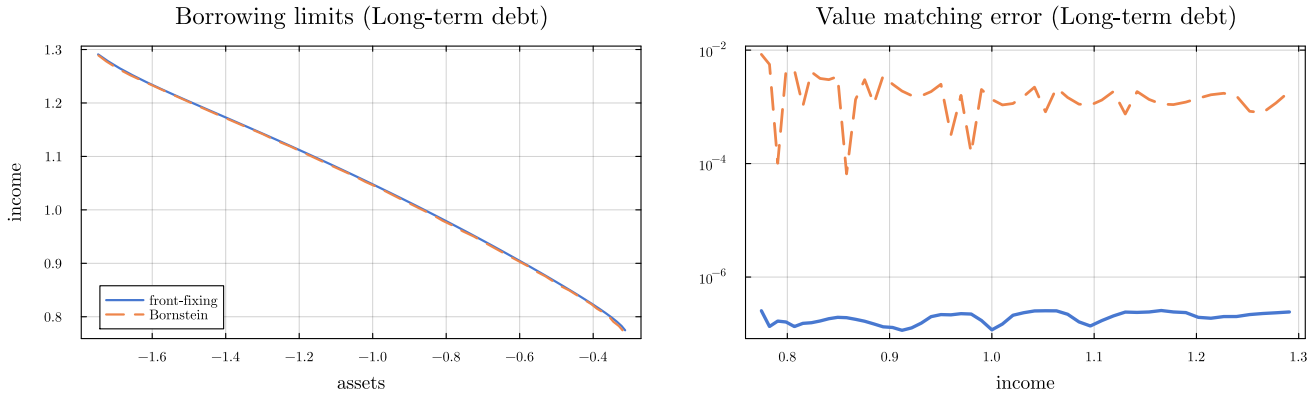


Figure 4: Comparison of *front-fixing* method with *trial* method of Bornstein (2020) for long-term debt model.

In contrast to the short-term debt model the policy functions between the different models diverge slightly as shown in Figure 5. This is mainly driven by a more accurate solution to the bond pricing equation relative to Bornstein (2020). In fact, Figure 6 shows that the residual of the bond pricing equation is non-negligible in Bornstein (2020) for a region in the state space. What is more, the region with the largest residuals coincides with where the probability of default starts to increase. This underscores the importance of using a fully implicit method that uses the information from the full Jacobian of the system for solving the long-term debt model.

3.3 Transition dynamics

Next, I show how to apply the front fixing technique to computing transition dynamics or more generally to a time-dependent problem. I consider the simplest model with short-term debt from Section 2. The borrowing limit is now defined by the mapping $(y,t) \mapsto \underline{a}(y,t)$. The system of the non-

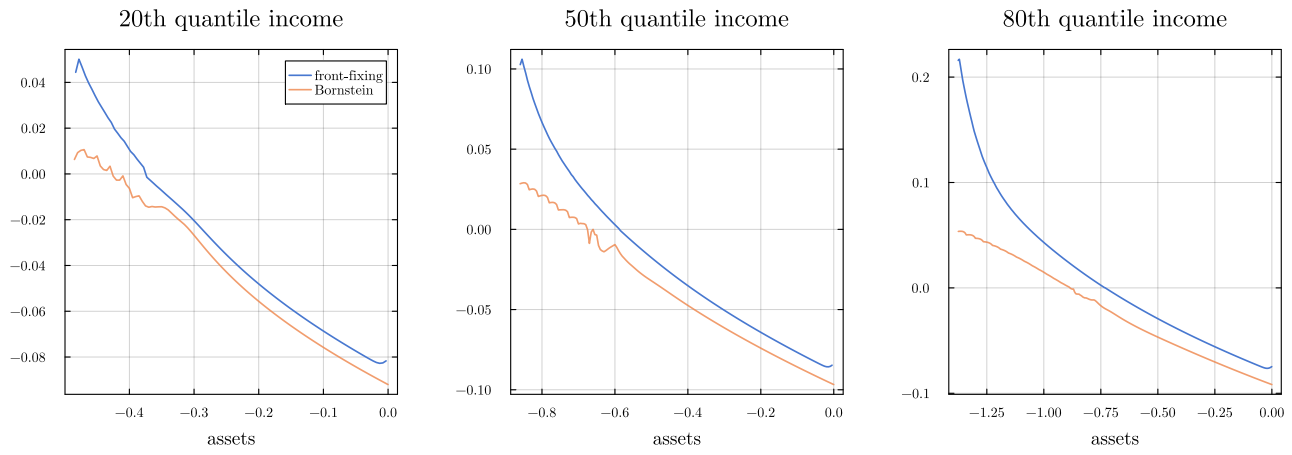


Figure 5: Comparison of *front-fixing* method with *trial* method of Bornstein (2020) for long-term debt model. Savings policy functions.

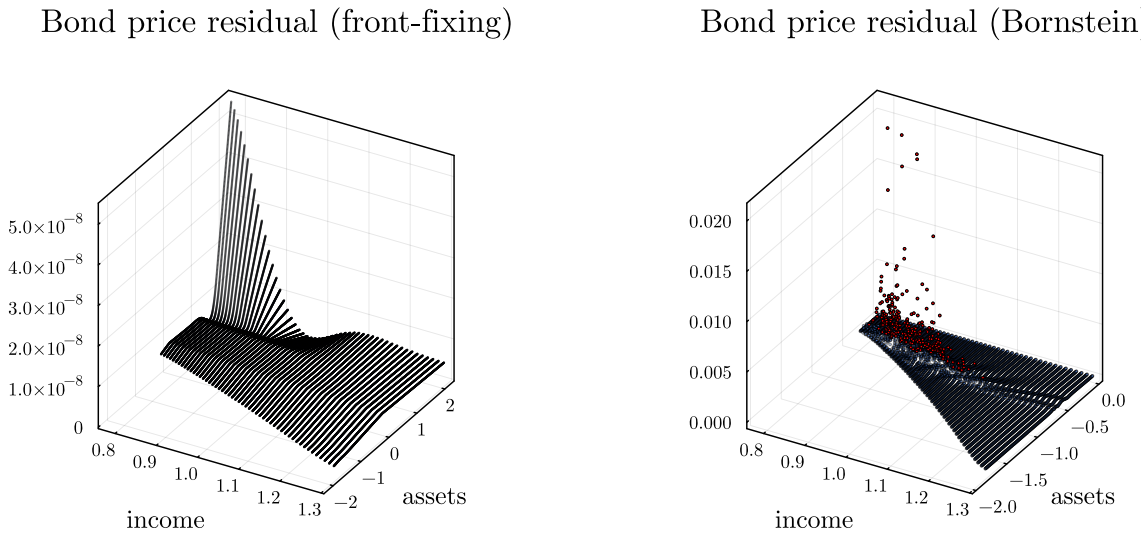


Figure 6: Comparison of *front-fixing* method with *trial* method of Bornstein (2020) for long-term debt model. Residual in bond pricing equation.

transformed HJBs is

$$\begin{aligned}\rho v_t(a,y) = \max\Big\{&\rho w(y),\ \max_c\ u(c) + \partial_a v_t(a,y)\left[y - c + r_t(a,y)a\right] \\ &+ \lambda_y \int_0^\infty \left[v_t(a,y') - v_t(a,y)\right] f(y,y')dy' \\ &+ \dot{v}_t(a,y)\Big\}\end{aligned} \tag{3.9}$$

$$\begin{aligned}\rho w_t(y) = u(y - \phi(y)) &+ \lambda_y \int_0^\infty \left[w_t(y') - w_t(y)\right] f(y,y')dy' \\ &+ \lambda_D\left[v_t(0,y) - w_t(y)\right] + \dot{w}_t(y)\end{aligned} \tag{3.10}$$

$$v_t(\underline{a}(y,t),y) = w_t(y) \tag{3.11}$$

subject to the income-specific borrowing (state) constraint

$$a \geq \underline{a}(y,t), \qquad \forall (y,t) \in \mathbb{R}^2_+$$

and the endogenous interest rate schedule reflecting default

$$r_t(a,y) = r_f + \lambda_y \int_0^{+\infty} D_t(a,y') f(y,y') dy', \tag{3.12}$$

Change of variables. Again we guess a function $\varphi_t\left(a - \underline{a}(y,t), y\right) := v_t(a,y)$ such that

$$\begin{aligned} \rho\varphi_t(\tilde{a},y) = \max_c u(c) &+ \partial_{\tilde{a}}\varphi_t(\tilde{a},y)\left[y - c + r_t(\tilde{a},y)(\tilde{a} + \underline{a}(y,t))\right] \\ &+ \lambda_y \int \left[\varphi_t(\tilde{a} + \underline{a}(y,t) - \underline{a}(y',t), y') - \varphi_t(\tilde{a},y)\right] f(y,y') dy \\ &+ \underbrace{\dot{\varphi}_t(\tilde{a},y) - \partial_{\tilde{a}}\varphi_t(\tilde{a},y)\dot{\underline{a}}(y,t)}_{= \frac{\partial}{\partial t}\varphi_t(a - \underline{a}(y,t), y)} \end{aligned} \tag{3.13}$$

Collecting terms, in the no default region $\tilde{a} \geq \underline{a}(y,t)$

$$\begin{aligned} \rho\varphi_t(\tilde{a},y) = \max_c u(c) &+ \partial_{\tilde{a}}\varphi_t(\tilde{a},y)\left[y - c + r_t(a,y)(\tilde{a} + \underline{a}(y,t)) - \dot{\underline{a}}(y,t)\right] \\ &+ \lambda_y \int \left[\varphi_t(\tilde{a} + \underline{a}(y,t) - \underline{a}(y',t), y') - \varphi(\tilde{a},y)\right] f(y,y') dy \\ &+ \dot{\varphi}_t(\tilde{a},y) \end{aligned} \tag{3.14}$$

Numerical example: Suppose that the rate at which sovereigns re-join capital markets transitions to a new, higher value. Specifically, $\dot{\lambda_D}(t) = 0.015 \cdot \left(\lambda_D(t) - \lambda_D^H\right)$.

Discretize grids $\tilde{a}$ and y as well as the stochastic process over y'. Compute initial and terminal steady states. Initialize a guess for the time path $\varphi(\tilde{a}, y, t)$ and $w(y, t)$ (e.g. terminal steady state). Set iteration initial time t to T.

for *n in 1:N* **do**

1. Given $\varphi(\bullet, \bullet, t)$ and $w(\bullet, t)$ find $\varphi(\bullet, \bullet, t-\Delta)$ and $w(\bullet, t-\Delta)$ that jointly solves (3.14) and the value matching condition (3.11) evaluated at $t-\Delta$. A Broyden method with occasional Jacobian resetting works well in practice.
2. Set $t = t-\Delta$. If $t = 0$, stop. Otherwise, continue iterating backwards.

end

Algorithm 3: Solving transition dynamics in short-term debt model.

The time path of the borrowing limit along the transition path is shown in Figure 7.

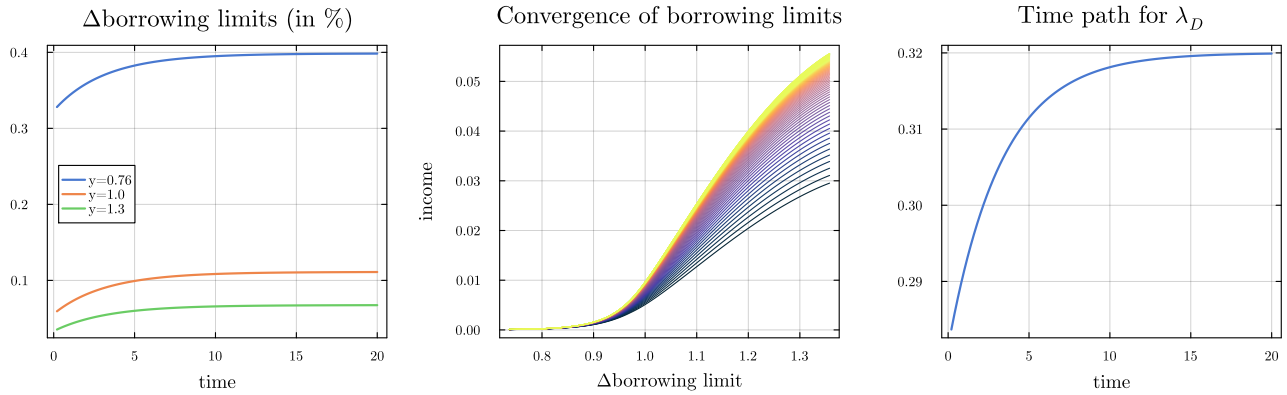


Figure 7: Transition dynamics to change in λ_D.

4 Conclusion

I propose a front-fixing framework to solving sovereign default models with endogenous borrowing constraints and borrowing costs. I apply the method to both short-term and long-term debt models. The front-fixing approach keeps the computational domain fixed which simplifies the problem while treating the borrowing constraints as continuous variables allows for greater accuracy. I show how Jacobians and Jacobian vector products can be efficiently computed by exploiting the structure of the problem and the chain rule. I also sketch possible extensions of the framework to models with diffusion and transition dynamics.

References

Achdou, Y., Han, J., Lasry, J.-M., Lions, P.-L. & Moll, B. (2021), 'Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach', *The Review of Economic Studies* **89**(1), 45–86.

Arellano, C. (2008), 'Default risk and income fluctuations in emerging economies', *The American Economic Review* **98**(3), 690–712.

Barles, G. & Souganidis, P. (1991), 'Convergence of approximation schemes for fully nonlinear second order equations', *Asymptotic Analysis* **4**(3), 271–283.

Bornstein, G. (2020), 'A continuous-time model of sovereign debt', *Journal of Economic Dynamics and Control* .

Coffey, T. S., Kelley, C. T. & Keyes, D. E. (2003), 'Pseudotransient continuation and differential-algebraic equations', *SIAM Journal on Scientific Computing* **25**(2), 553–569.

Company, R., Egorova, V. N. & Jódar, L. (2021), 'A front-fixing etd numerical method for solving jump–diffusion american option pricing problems', *Mathematics and Computers in Simulation* **189**, 69–84. MATCOM Special Issue: Modelling 2019: The 6th International Conference on Mathematical Modelling and Computational Methods in Applied Sciences and Engineering.

Crank, J. (1957), 'Two methods for the numerical solution of moving-boundary problems in diffusion and heat flow', *The Quarterly Journal of Mechanics and Applied Mathematics* **10**(2), 220–231.

Duffy, D. J. (2006), *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*, Wiley.
URL: *http://dx.doi.org/10.1002/9781118673447*

Fazio, R., Insana, A. & Jannelli, A. (2021), 'A front-fixing implicit finite difference method for the american put options model', *Mathematical and Computational Applications* **26**.

Gomez, M. (2024), Wealth inequality and asset prices.

Heidari, S. & Azari, H. (2017), 'A front-fixing finite element method for pricing american options under regime-switching jump-diffusion models', *Computational and Applied Mathematics* **37**(3).

Hurtado, S., Nuño, G. & Thomas, C. (2022), 'Monetary Policy and Sovereign Debt Sustainability', *Journal of the European Economic Association* **21**(1), 293–325.

Landau, H. G. (1950), 'Heat conduction in a melting solid', *Quarterly of Applied Mathematics* **8**, 81–94.

Nielsen, B., Skavhaug, O. & Tveito, A. (2001), 'Penalty and front-fixing methods for the numerical solution of american option problems', *Journal of Computational Finance* **5**.

Wu, L. & Kwok, Y. K. (1997), 'A front-fixing finite difference method for the american option pricing problem', *The Journal of Financial Engineering* **6**(2).

Zhu, Y.-l., Chen, B.-m., Ren, H. & Xu, H. (2003), *Advances in Computational Mathematics* **19**(1/3), 147–158.

A An analytical example

The front-fixing approach essentially is the numerical implementation of the analytical approach to solving optimal stopping time problems. To illustrate this, consider the following simple stopping time problem. A firm has the option to shut down and sell its asset at any time at a scrap value S_0. Suppose that entrepreneurs have utility $u(x) = \log(x)$ over profits and that profits follow a geometric Brownian motion

$$\frac{dX_t}{X_t} = -\mu dt + \sigma dB_t. \tag{A.1}$$

The optimal stopping time problem is

$$\begin{aligned} r\,v(x) &= \log x - v'(x)\mu x + \frac{1}{2}\sigma^2 x^2 v''(x), \quad && x > \underline{x} \\ v(x) &= S_0, && x \leq \underline{x}. \end{aligned} \tag{A.2}$$

The familiar boundary conditions are

$$\lim_{x \downarrow \underline{x}} v(x) = S_0 \qquad \text{(Value matching)}$$

$$\lim_{x \downarrow \underline{x}} v'(x) = 0 \qquad \text{(Smooth pasting)}$$

We call this a free boundary problem in the sense that the boundary $\underline{x}$ is unknown and must be solved for. Specifically, the boundary must satisfy the boundary conditions of value matching and smooth pasting. To obtain a solution we proceed in a series of steps:

1. **Change of variable:** We transform the problem into a fixed domain and define the new function $\varphi\left(\frac{x}{\underline{x}}\right) = v(x)$ such that, denoting by $\tilde{x} := \frac{x}{\underline{x}}$,

$$\begin{aligned} r\varphi(\tilde{x}) &= \log(\underline{x}) + \log(\tilde{x}) - \varphi'(\tilde{x})\mu\tilde{x} + \frac{1}{2}\sigma^2\tilde{x}^2\varphi''(\tilde{x}), \quad && \tilde{x} > 1 \\ \varphi(\tilde{x}) &= S_0, && x \leq 1. \end{aligned} \tag{A.3}$$

$$\varphi(1) = S_0, \qquad \varphi'(1) = 0 \tag{A.4}$$

2. **Solve the ODE:** The homogeneous solution to the ODE is $c_1 x^{z_1} + c_2 x^{z_2}$, where $z_{1,2} = \frac{\hat{\mu} \pm \sqrt{\hat{\mu}^2 + 2\sigma^2 r}}{\sigma^2}$ and $\hat{\mu} := \mu + \frac{1}{2}\sigma^2$. By standard arguments we rule out the explosive root for the homogeneous

solution and denote $z = -\frac{\hat{\mu}-\sqrt{\hat{\mu}^2+2\sigma^2 r}}{\sigma^2}$ such that the homogeneous part of the solution is $c_1 x^{-z}$.

To find the particular solution we guess $\varphi(\tilde{x}) = A + B\log(\tilde{x})$ and plug this into the ODE. This gives us the particular solution as a function of the free boundary

$$A(\underline{x}) = \frac{1}{r}\left(\log \underline{x} - \frac{\hat{\mu}}{r}\right), \qquad B = \frac{1}{r}. \tag{A.5}$$

Next, we impose smooth pasting. That is, we require that $\varphi'(1) = 0$ which gives us the boundary condition

$$-zc + \frac{1}{r} = 0, \qquad \Longrightarrow c = \frac{1}{rz}. \tag{A.6}$$

Note that this is what we would do in a numerical solution. We impose one boundary condition and solve for the value function that is parameterised by the free boundary $\underline{x}$ yet to be solved.

3. **Solve for the boundary:** We now use the value matching condition. This requires that $\varphi(1) = S_0$ which gives us

$$A(\underline{x}) = S_0 - \frac{1}{rz}, \qquad \Longrightarrow \underline{x} = \exp\left\{rS_0 + \frac{\mu + \frac{1}{2}\sigma^2}{r} - \frac{1}{z}\right\}. \tag{A.7}$$

Putting all components together we observe that

$$\varphi(\tilde{x}) = \begin{cases} \left(S_0 - \frac{1}{rz}\right) + \frac{1}{rz}\tilde{x}^{-z} + \log(\tilde{x}) & \tilde{x} > 1, \\ S_0 & \tilde{x} \leq 1. \end{cases}; \qquad z = -\frac{\mu + \frac{1}{2}\sigma^2 - \sqrt{(\mu + \frac{1}{2}\sigma^2)^2 + 2\sigma^2 r}}{\sigma^2}. \tag{A.8}$$

In the numerical solution we would proceed in a similar manner. The only difference is that we would need to solve the PDE in 2. numerically using a monotone finite difference scheme and solve 3. using a root finding algorithm. We also cannot generically impose both smooth pasting and value matching as boundary conditions of the finite difference operators and thus we would need to proceed in a similar fashion, by imposing one boundary condition and using the other boundary condition to solve for the free boundary. It does not make a difference which of the two is imposed first.

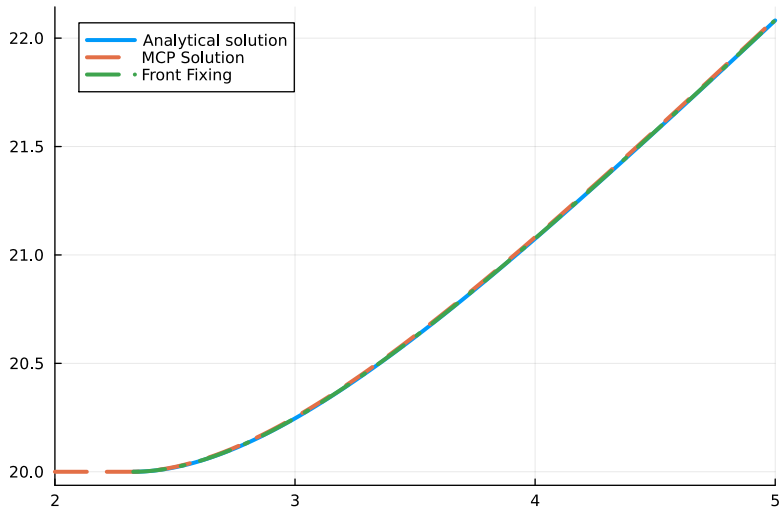


Figure 8: Price of option for different solution methods: (i) analytical solution (ii) mixed complementarity problem (iii) front-fixing.

B Derivations for change of variable

Plugging our change of variable $\varphi\left(a-\underline{a}(y),y\right):=v(a,y)$ into (2.1) we get for all $a\geq\underline{a}(y)$

$$\begin{aligned}\rho\varphi\left(a-\underline{a}(y),y\right)=\max_{c}u(c)+\frac{\partial}{\partial a}\varphi\left(a-\underline{a}(y),y\right)\left[y-c+r(a,y)a\right]\\+\lambda\int_{0}^{+\infty}\left[\varphi\left(a-\underline{a}(y'),y'\right)-\varphi\left(a-\underline{a}(y),y\right)\right]f(y,y')dy'\end{aligned}\tag{B.1}$$

Using the chain rule, or definition for $\tilde{a}=a-\underline{a}(y)$ and conjecturing denoting $r(a,y)=r(\tilde{a},y)$ we get

$$\begin{aligned}\rho\varphi\left(\tilde{a},y\right)=\max_{c}u(c)+\partial_{\tilde{a}}\varphi\left(\tilde{a},y\right)\left[y-c+r(\tilde{a},y)(a+\tilde{a})\right]\\+\lambda\int_{0}^{+\infty}\left[\varphi\left(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y'\right)-\varphi\left(\tilde{a},y\right)\right]f(y,y')dy'\end{aligned}\tag{B.2}$$

C Numerical details

C.1 Finite difference scheme

The general numerical scheme to solve PDEs largely follows Achdou et al. (2021). Consider the HJB for a solvent sovereign

$$\rho\varphi(\tilde{a},y)=\max_{c}u(c)+L[\varphi](\tilde{a},y,c;\underline{a}(y))+M[\varphi](\tilde{a},y;\underline{a}),\tag{C.1}$$

where the operators are defined as $L[\varphi](\tilde{a},y,c;\underline{a}(y)) := \partial_{\tilde{a}}\varphi(\tilde{a},y)\left[y + r(\tilde{a},y)(\tilde{a}+\underline{a}(y)) - c\right]$ and $M[\varphi](\tilde{a},y;\underline{a}) = \lambda_y \int_0^{+\infty}\left[\varphi(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y') - \varphi(\tilde{a},y)\right] f(y,y')dy'$.

We compute the PDE numerically using finite difference stencils of these operators. Using these stencils the PDE boils down to

$$\left(\rho I - L(c^*;\underline{a}) - M(\underline{a})\right)\varphi = u(c^*), \tag{C.2}$$

where c^* is the optimal policy function. In the short-term debt version of the model this boils down to a simple matrix inversion.[6] More generally, e.g. in the long-term debt version, we need to solve a fully non-linear system of equations for each point in time t. We can employ a standard Newton algorithm. We then solve for φ as

$$\mathscr{J}^n d\varphi = -\left[\left(\rho I - L^n(c^*;\underline{a}) - M^n(\underline{a})\right)\varphi^n - u(c^n)\right], \qquad \varphi^{n+1} = \varphi^n + d\varphi. \tag{C.3}$$

For stability purposes we may also use fictional time-stepping, which in the above notation would show up as a dampened Newton method akin to the well-known Levenberg-Marquardt algorithm. The advantage of writing the equation in the above way is that we may use matrix-free Krylov methods to solve for $d\varphi$ (e.g. BICGSTAB or GMRES). This can be particularly useful if the dimension of the state space is large or if the jump process is very dense. Instead of computing the factorisation of the Jacobian we only need to compute matrix-vector products which can be much more efficient. Furthermore, we can reuse previous iterations to use as initial guess for Krylov subspace based solvers.

C.2 Implicit-explicit scheme

The above scheme corresponds to a so-called fully implicit scheme. These schemes tend to be most stable – even for large time steps. However, they are also computationally more expensive. In particular, if the jump matrix is very dense this can be a bottleneck in the code. An alternative scheme that is popular in option pricing is the IMEX (implicit-explicit) scheme which treats the differential operator implicitly and the jump operator explicitly. For example, consider the HJB

$$\rho\varphi_t(\tilde{a},y) = \max_c u(c) + L[\varphi_t](\tilde{a},y,c;\underline{a}(y)) + M[\varphi_t](\tilde{a},y;\underline{a}) + \partial_t\varphi_t(\tilde{a},y). \tag{C.4}$$

[6] Potential non-linear effects cancel due to the envelope condition.

The fully implicit method uses the discretisation

$$\rho\varphi_t(\tilde{a},y) = \max_c u(c) + L_t[\varphi_t](\tilde{a},y,c;\underline{a}(y)) + M_t[\varphi_t](\tilde{a},y;\underline{a}) + \frac{\varphi_{t+\Delta}(\tilde{a},y)-\varphi_t(\tilde{a},y)}{\Delta}. \tag{C.5}$$

We solve for φ_t as $\varphi_t = \left(I+\Delta\left[\rho I - (L_{t+\Delta}+M_{t+\Delta})\right]\right)^{-1}\left[\Delta u(c^*_{t+\Delta})+\varphi_{t+\Delta}\right]$.

The implicit-explicit scheme uses the discretisation

$$\rho\varphi_t(\tilde{a},y) = \max_c u(c) + L_{t+\Delta}[\varphi_t](\tilde{a},y,c;\underline{a}(y)) + M_{t+\Delta}[\varphi_{t+\Delta}](\tilde{a},y;\underline{a}) + \frac{\varphi_{t+\Delta}(\tilde{a},y)-\varphi_t(\tilde{a},y)}{\Delta}. \tag{C.6}$$

We solve for φ_t as $\varphi_t = \left(I+\Delta\left[\rho I - L_{t+\Delta}\right]\right)^{-1}\left[\Delta u(c^*_{t+\Delta}) + (I+\Delta M_{t+\Delta})\varphi_{t+\Delta}\right]$.[7] We may also replace φ_t in the jump term with an extrapolation $\varphi_t \approx \varphi_{t+\Delta} + \frac{\varphi_{t+\Delta}-\varphi_{t+2\Delta}}{\Delta}\cdot\Delta = 2\varphi_{t+\Delta}-\varphi_{t+2\Delta}$.

C.3 IMEX-BDF2

In the above we have simply used $\partial_t\varphi_t(\tilde{a},y) \approx \frac{\varphi_{t+\Delta}(\tilde{a},y)-\varphi_t(\tilde{a},y)}{\Delta}$. Different discretisations are possible. For example the BDF2 scheme[8] uses $\partial_t\varphi_t(\tilde{a},y) \approx -\frac{3\varphi_{t-\Delta}(\tilde{a},y)-4\varphi_t(\tilde{a},y)+\varphi_{t+\Delta}(\tilde{a},y)}{2\Delta}$ (recall that we're going backwards in time). Further, the IMEX-BDF2 scheme uses an extrapolation step for the jump term to approximate $\varphi_{t-\Delta} \approx \varphi_t + \frac{\varphi_t-\varphi_{t+\Delta}}{\Delta}\Delta = 2\varphi_t - \varphi_{t+\Delta}$. Hence, the discretised HJB becomes

$$\rho\varphi_{t-\Delta}(\tilde{a},y) = \max_c u(c) + L_t[\varphi_{t-\Delta}](\tilde{a},y,c;\underline{a}(y)) + M_t[2\varphi_t-\varphi_{t+\Delta}](\tilde{a},y;\underline{a}) - \frac{3\varphi_{t-\Delta}(\tilde{a},y)-4\varphi_t(\tilde{a},y)+\varphi_{t+\Delta}(\tilde{a},y)}{2\Delta}. \tag{C.7}$$

This can now be solved for $\varphi_{t-\Delta} = \left(I+2\Delta\left(\rho I - L_t\right)\right)^{-1}\left[2\Delta u(c^*_t) + 4(I+\Delta M_t)\varphi_t - (I+2\Delta M_t)\varphi_{t+\Delta}\right]$. It is straightforward to extend this method to other time integration schemes.

[7] In practice, if the operator $M[\varphi](\tilde{a},y;\underline{a}) = \lambda_y\left[\int_0^{+\infty}\varphi(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y')f(y,y')dy' - \varphi(\tilde{a},y)\right]$ it is useful to split the integral term and the other term. That is, denote $\tilde{M}[\varphi](\tilde{a},y;\underline{a}) = \lambda\int_0^{+\infty}\varphi(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y')f(y,y')dy'$ and the IMEX scheme reduces to solving $\varphi_t = \left(I+\Delta\left[(\rho+\lambda_y)I - L_{t+\Delta}\right]\right)^{-1}\left[\Delta u(c^*_{t+\Delta}) + \left(I+\Delta\tilde{M}_{t+\Delta}\right)\varphi_{t+\Delta}\right]$. This turns out to have nicer properties.

[8] See https://en.wikipedia.org/wiki/Backward_differentiation_formula.

C.4 Splitting computation of φ and w

The value matching condition (2.8) imposes $\varphi(0,y) = w(y)$. Define the set $\mathscr{A} = \{y' \in \mathbb{R}^+ : \tilde{a} + \underline{a}(y) - \underline{a}(y') \geq 0\}$. It follows that we can compute the integral term as follows

$$\int_0^{+\infty} \varphi(\tilde{a} + \underline{a}(y) - \underline{a}(y'))f(y,y')dy' = \int_{\mathscr{A}} \varphi(\tilde{a} + \underline{a}(y) - \underline{a}(y'))f(y,y')dy' + \int_{\mathscr{A}^c} w(y')f(y,y')dy' \tag{C.8}$$

$$= \int_{\mathscr{A}} \varphi(\tilde{a} + \underline{a}(y) - \underline{a}(y'))f(y,y')dy' + \int_{\mathscr{A}^c} \varphi(0,y')f(y,y')dy'. \tag{C.9}$$

It follows that we do not require any knowledge of the default value function w to compute the value function φ of being solvent. We can split the computation of the two value functions. Once we computed φ, we can easily back out w from (2.7) via a simple linear solve. What is more, we can precompute and store the factorisation of the Jacobian of the system.

C.5 Computing Jacobian

The solution to the sovereign defaul problem can be succinctly summarised as the triplet $(\varphi, w, \underline{a})$ that solves the following systems

$$H(\varphi, \underline{a}) = 0 \qquad \text{(HJB solvent)}$$

$$G(\varphi, w, \underline{a}) = 0 \qquad \text{(HJB default)}$$

$$F(\varphi, w) = 0 \qquad \text{(Value matching)}$$

The algorithm in the main text proceeds by fixing a guess for $\underline{a}$, solve for the tuple $(\varphi, \underline{a})$ and at last update $\underline{a}$ using the value matching equation. The updating of the default threshold $\underline{a}$ is done using a Newton method

$$\frac{dF^n}{d\underline{a}} \cdot d\underline{a} = -F^n, \qquad \underline{a}^{n+1} = \underline{a}^n + d\underline{a}. \tag{C.10}$$

Classic Newton: We wish to compute the Jacobian of the system $F(v,w) = 0$. This is a non-linear system nesting other non-linear functions. We can compute the Jacobian of the system using the chain rule together with the implicit function theorem. That is, we can write the Jacobian as

$$dF = \frac{\partial F}{\partial v} dv + \frac{\partial F}{\partial w} dw. \tag{C.11}$$

In turn we have $dw = -\left[\frac{\partial G}{\partial w}\right]^{-1}\left(\frac{\partial G}{\partial \varphi}d\varphi + \frac{\partial G}{\partial \underline{a}}d\underline{a}\right)$ and $d\varphi = -\left[\frac{\partial H}{\partial \varphi}\right]^{-1}\frac{\partial H}{\partial \underline{a}}d\underline{a}$. Hence, combining all parts[9]

$$dF = \left\{\left(\frac{\partial F}{\partial \varphi} + \frac{\partial F}{\partial w}\left[-\frac{\partial G}{\partial w}\right]^{-1}\frac{\partial G}{\partial \varphi}\right)\left[-\frac{\partial H}{\partial \varphi}\right]^{-1}\frac{\partial H}{\partial \underline{a}} + \frac{\partial F}{\partial w}\left[-\frac{\partial G}{\partial w}\right]^{-1}\frac{\partial G}{\partial \underline{a}}\right\}d\underline{a} \tag{C.12}$$

This may not seem like considerable progress at first, but thanks to the envelope theorem we can directly reuse the matrix that we constructed for the solution of the HJB. Further, $\frac{\partial G}{\partial w}$ and $\frac{\partial G}{\partial \varphi}$ corresponds similarly to the matrix that we have pre-computed for the linear solve of the value function in the default state.[10] It follows that the most complex objects (i.e. of the greatest dimensionality) can be reused directly from the inner loop. The remaining Jacobians can be computed efficiently using automatic differentiation.

There are potentially large efficiency gains in this computation by simply using clever bracketing. That is we want to solve the smallest systems first and then use the results to solve the larger systems.

Newton-Krylov: Computing the full Jacobian as outlined above may be inefficient if the dimensionality grows large. The main bottleneck is the linear solve $\left[-\frac{\partial H}{\partial \varphi}\right]^{-1}\frac{\partial H}{\partial \underline{a}}$. The number of linear solves required is that of the dimensionality of $\underline{a}$ while the size of each system to be solved corresponds to the dimension of φ. On the other hand, the Jacobian-Vector product $\frac{dv}{d\underline{a}}d\underline{a}$ only requires solving a single linear system $\left[-\frac{\partial H}{\partial \varphi}\right]^{-1}\left(\frac{\partial H}{\partial \underline{a}}d\underline{a}\right)$.

Specifically, we can proceed as follows using the following directional derivatives

$$\begin{aligned}
dv &= \left[-\frac{\partial H}{\partial \varphi}\right]^{-1}\lim_{t\to 0}\frac{H(v,\underline{a}+t\cdot d\underline{a}) - H(v,\underline{a})}{t} \\
dw &= \left[-\frac{\partial G}{\partial w}\right]^{-1}\left(\lim_{t\to 0}\frac{G(w,v+t\cdot dv,\underline{a}) - G(w,v,\underline{a})}{t} + \lim_{t\to 0}\frac{G(w,v,\underline{a}+t\cdot d\underline{a}) - G(w,v,\underline{a})}{t}\right) \\
dF &= \lim_{t\to 0}\frac{F(v+t\cdot dv,w)}{t} + \lim_{t\to 0}\frac{F(v,w+t\cdot dw)}{t}
\end{aligned} \tag{C.13}$$

With recursive computation of the Jacobian-Vector products we can compute dF very efficiently. This

[9] Another approach is to use the implicit function theorem on the larger system and note that

$$\begin{bmatrix} G_\varphi & G_w \\ F_\varphi & F_w \end{bmatrix}\begin{bmatrix} d\varphi \\ dw \end{bmatrix} = -\begin{bmatrix} G_{\underline{a}} \\ F_{\underline{a}} \end{bmatrix}d\underline{a}.$$

However, for efficiency reasons it is better to use the chain rule, especially when computin the Jacobian-Vector products. When dealing with the Jacobian-Vector products we in fact do not need to compute all elements of the Jacobian.

[10] This is the straightforward result from the fact that the Jacobian of a linear system is equal to the linear map.

allows us to use e.g. GMRES or BicGstab to find the Newton updating step or equivalently the updating direction $d\underline{a}$. This involves an iterative approach. That is, guess a direction $d\underline{a}$ solve for the directional derivative dF and update until $dF = -F^n$.

Finite difference JVP: The directional derivative can also be computed using finite differences. Some existing non-linear solvers do this automatically such as the KINSOL solver from Sundials.[11] This is the easiest for the user to implement but introduces numerical error.

C.6 Analytical Jacobian

Consider the HJB and take the derivative with respect to $\underline{a}(\hat{y})$. Denote $\psi(a,y,\underline{a}(\hat{y})) = \frac{\partial}{\partial \underline{a}(\hat{y})}\varphi(a,y)$. It can be shown that

$$\begin{aligned}\psi(a,y,\underline{a}(\hat{y})) = u(c^*(a,y)) + u'(c^*(a,y))\left[\mathbf{1}\{y=\hat{y}\} + \frac{\partial r}{\partial \underline{a}(\hat{y})}(a,y)\right] \\ + \partial_a \psi(a,y,\underline{\hat{y}})\dot{a} \\ + \lambda_y \int_0^{+\infty} \left[\psi(a+\underline{a}(y)-\underline{a}(y'),y',\underline{a}(\hat{y})) - \psi(a,y,\underline{a}(\hat{y}))\right] f(y,y')dy' \\ + \lambda_y \int_0^{+\infty} u'(c^*(a+\underline{a}(y)-\underline{a}(y'),y'))\left[\mathbf{1}\{\hat{y}=y\} - \mathbf{1}\{\hat{y}=y'\}\right] f(y,y')dy'\end{aligned} \tag{C.14}$$

That is, the Jacobian satisfies their own PDE. However, numerically it is more stable to work with the discretised system directly and not necessarily with the ideal system.

C.7 Updating without a Jacobian

The following algorithm works as well, though it is much less efficient. suggests the following algorithm.

[11] For more details see https://sundials.readthedocs.io/en/latest/kinsol/Mathematics_link.html. For a Julia implementation see https://docs.sciml.ai/NonlinearSolve/stable/solvers/nonlinear_system_solvers/.

Discretize grids a and y as well as the stochastic process over y'. Initialize an arbitrary guess for $\varphi(\tilde{a}, y)$ and $w(y)$.

while $||\varphi(0, y) - w(y)||_\infty > tol$ **do**

1. Solve for φ and w given a guess for $\underline{a}$.
2. Compute the value matching error. If the error is sufficiently small quit, otherwise proceed to the next step.
3. Using the HJB for the default value function find the update of the default threshold $\underline{a}$ by solving

$$\lambda_D \varphi^n(-\underline{a}^{n+1}(y), y) = (\rho + \lambda_y + \lambda_D) w^n(y) - \lambda_y \int_0^{+\infty} w^n(y') f(y, y') dy - u(y - \phi(y)), \quad \forall y. \tag{C.15}$$

We can use linear interpolation to compute the inverse of φ and back out a new update for $\underline{a}(y)$.

end

Algorithm 4: Iterative algorithm for short-term debt model.

C.8 Long-term debt

Since $1/q(a, y)$ appears in the asset drift the joint system of PDEs is highly non-linear. To solve this system efficiently I use a fully implicit IMEX-BDF2 scheme. At each instant t I solve the non-linear system of equations using a Newton solver, using the fact that the Jacobians are highly sparse to speed up computations. This can be achieved via matrix coloring and sparse automatic differentiation.

Consider the joint discretised system which we solve for φ and Q at each time step. Old values are

denoted with superscript n and the new values with superscript $n+1$.

$$\begin{bmatrix} \rho\varphi^{n+1}(\tilde{a},y) - \left\{ u(c^{n+1}) + \partial_{\tilde{a}}\varphi^{n+1}(\tilde{a},y)\left[\frac{y-c^{n+1}}{Q^{n+1}(\tilde{a},y)} + \left(\frac{z+\lambda_b}{Q^{n+1}(\tilde{a},y)} - \lambda_b\right)(\tilde{a}+\underline{a}(y))\right] \right. \\ \left. + \lambda_y \int_0^{+\infty} \left[\varphi^n\left(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y'\right) - \varphi^{n+1}(\tilde{a},y)\right] f(y,y')dy' \right\} \\ - \frac{\varphi^n(\tilde{a},y) - \varphi^{n+1}(\tilde{a},y)}{\Delta} \\ r_f Q^{n+1}(\tilde{a},y) - z + \lambda_b(1-Q^{n+1}(\tilde{a},y)) - \partial_{\tilde{a}}Q^{n+1}(\tilde{a},y)\left[\frac{y-c^{n+1}(\tilde{a},y)}{Q^{n+1}(\tilde{a},y)} + \left(\frac{z+\lambda_b}{Q^{n+1}(\tilde{a},y)} - \lambda_b\right)(\tilde{a}+\underline{a}(y))\right] \\ - \lambda_y \int_0^{\infty} \left[Q^n\left(\tilde{a}+\underline{a}(y)-\underline{a}(y'),y'\right) - Q^n(\tilde{a},y)\right] f(y,y')dy' \\ - \frac{Q^n(\tilde{a},y) - Q^{n+1}(\tilde{a},y)}{\Delta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{C.16}$$

The algorithm proceeds as follows:

while $||\varphi(0,y) - w(y)||_\infty > tol$ **do**

1. For a given guess of φ^n and Q^n solve the above system (C.16) using a Newton method for a given time step Δ. For Δ sufficiently small the system is guaranteed to converge. If the Newton solver stagnates reduce the time step Δ.
2. Once the system has been solved for φ^{n+1} and Q^{n+1}, check for convergence in $\frac{\varphi^n(\tilde{a},y)-\varphi^{n+1}(\tilde{a},y)}{\Delta}$ and $\frac{Q^n(\tilde{a},y)-Q^{n+1}(\tilde{a},y)}{\Delta}$. If the error is sufficiently small proceed to the next time step, otherwise repeat the above step.
3. Once the system has been solved for φ^{n+1} and Q^{n+1} we can solve for w. This solves the inner loop. We check the error of the value matching condition. If the error is sufficiently small we are done. If not proceed to the next step.
4. With functions φ^{n+1}, Q^{n+1} and w^{n+1} we compute the Jacobian and/or Jacobian vector product of the system and find an update for the borrowing limits $\underline{a}^{n+1}$.

end

Algorithm 5: Details long-term debt algorithm.

D General change of variable

D.1 Derivative Mapping

Consider the following change of variable $f(g(x,y),h(x,y))$. Then,

$$\begin{aligned}\nabla f(g(x,y),h(x,y)) &= \begin{bmatrix} f_1 g_x + f_2 h_x & f_1 g_y + f_2 h_y \end{bmatrix} \\ &= \begin{bmatrix} f_1 & f_2 \end{bmatrix} \begin{bmatrix} g_x & g_y \\ h_x & h_y \end{bmatrix}.\end{aligned} \tag{D.1}$$

The Hessian is given by

$$\begin{aligned}&\nabla^2 f(g(x,y),h(x,y)) = \\ &\begin{bmatrix} f_{11}g_x^2 + 2f_{12}g_x h_x + f_{22}h_x^2 + f_1 g_{xx} + f_2 h_{xx} & f_{11}g_x g_y + f_{12}(g_x h_y + g_y h_x) + f_{22}h_x h_y + f_1 g_{xy} + f_2 h_{xy} \\ f_{11}g_x g_y + f_{12}(g_x h_y + g_y h_x) + f_{22}h_x h_y + f_1 g_{xy} + f_2 h_{xy} & f_{11}g_y^2 + 2f_{12}g_y h_y + f_{22}h_y^2 + f_1 g_{yy} + f_2 h_{yy} \end{bmatrix} \\ &= \begin{bmatrix} g_x & g_y \\ h_x & h_y \end{bmatrix}^T \begin{bmatrix} f_{11} & f_{12} \\ f_{12} & f_{22} \end{bmatrix} \begin{bmatrix} g_x & g_y \\ g_x & h_y \end{bmatrix} + f_1 \begin{bmatrix} g_{xx} & g_{xy} \\ g_{xy} & g_{yy} \end{bmatrix} + f_2 \begin{bmatrix} h_{xx} & h_{xy} \\ h_{xy} & h_{yy} \end{bmatrix} \\ &= \begin{bmatrix} g_x & g_y \\ h_x & h_y \end{bmatrix}^T \begin{bmatrix} f_{11} & f_{12} \\ f_{12} & f_{22} \end{bmatrix} \begin{bmatrix} g_x & g_y \\ h_x & h_y \end{bmatrix} + \begin{bmatrix} g_{xx} & g_{xy} & h_{xx} & h_{xy} \\ g_{xy} & g_{yy} & h_{xy} & h_{yy} \end{bmatrix} \left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \otimes I_2 \right).\end{aligned} \tag{D.2}$$

D.2 Simplification

Consider a special case of the above in which $g_x = h_y = 1$, $h_x = 0$, $g_{xx} = g_{xy} = h_{xx} = h_{xy} = 0$ and $\Sigma = \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2 \end{bmatrix}$. This implies that $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & g_{yy} & 0 & 0 \end{bmatrix} \left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \otimes I_2 \right) = \begin{bmatrix} 0 & 0 \\ 0 & f_1 g_{yy} \end{bmatrix}$. Then,

$$
\begin{aligned}
\operatorname{trace}\left\{\Sigma \nabla^2 f(g(x,y),h(x,y))\right\} &= \operatorname{trace}\left\{ \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ g_y & 1 \end{bmatrix} \nabla^2 f \begin{bmatrix} 1 & g_y \\ 0 & 1 \end{bmatrix} \right\} + \operatorname{trace}\left\{ \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & f_1 g_{yy} \end{bmatrix} \right\} \\
&= \sigma^2 \operatorname{trace}\left\{ \begin{bmatrix} 1 & g_y \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ g_y & 1 \end{bmatrix} \nabla^2 f \right\} + \sigma^2 f_1 g_{yy} \\
&= \sigma^2 \operatorname{trace}\left\{ \begin{bmatrix} 0 & g_y \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ g_y & 1 \end{bmatrix} \nabla^2 f \right\} + \sigma^2 f_1 g_{yy} \\
&= \sigma^2 \operatorname{trace}\left\{ \begin{bmatrix} g_y^2 & g_y \\ g_y & 1 \end{bmatrix} \nabla^2 f \right\} + \sigma^2 f_1 g_{yy} \\
&= \sigma^2 \begin{bmatrix} g_y & 1 \end{bmatrix} \nabla^2 f \begin{bmatrix} g_y \\ 1 \end{bmatrix} + \sigma^2 f_1 g_{yy}.
\end{aligned}
\tag{D.3}
$$