A sparse spectral method for Volterra integral equations

Timon S. Gutleb, Sheehan Olver

Imperial College London

28th Biennial Numerical Analysis Conference, June 2019

Overview

Introduction

Volterra integral equations

Define the *Volterra integral operator*

$$(\mathcal{V}_K u)(x) := \int_0^{\ell(x)} K(x,y)u(y)\mathrm{d}y,$$

where $K(x,y)$ is called the kernel, $u(y)$ is a given function of one variable. The limits of integration are either

$$\ell(x) = x \qquad \text{or} \qquad \ell(x) = 1 - x.$$

We introduce a sparse spectral method to find numerical approximations to the solution of Volterra integral equations of the first and second kind, i.e. to find u satisfying

$$\mathcal{V}_K u = g \qquad \text{or} \qquad (I + \mathcal{V}_K)u = g.$$

Volterra integral equations

Define the *Volterra integral operator*

$$(\mathcal{V}_K u)(x) := \int_0^{\ell(x)} K(x,y)u(y)\mathrm{d}y,$$

where $K(x,y)$ is called the kernel, $u(y)$ is a given function of one variable. The limits of integration are either

$$\ell(x) = x \qquad \text{or} \qquad \ell(x) = 1 - x.$$

We introduce a sparse spectral method to find numerical approximations to the solution of Volterra integral equations of the first and second kind, i.e. to find u satisfying

$$\mathcal{V}_K u = g \qquad \text{or} \qquad (I + \mathcal{V}_K)u = g.$$

Function approximation with orth. polynomials

We expand functions using a complete basis of orthogonal polynomials:

$$f(x) = \sum_{n=0}^{\infty} P_n(x) f_n = \mathbf{P}(x)^{\mathsf{T}} \mathbf{f},$$

where

$$\mathbf{P}(x) := \begin{pmatrix} P_0(x) \\ P_1(x) \\ \vdots \end{pmatrix}, \quad \mathbf{f} := \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix}.$$

This works analogously for bivariate orthogonal polynomials

$$f(x,y) = \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} P_{n,k}(x,y) f_{n,k}.$$

Function approximation with orth. polynomials

We expand functions using a complete basis of orthogonal polynomials:

$$f(x) = \sum_{n=0}^{\infty} P_n(x) f_n = \mathbf{P}(x)^{\mathsf{T}} \mathbf{f},$$

where

$$\mathbf{P}(x) := \begin{pmatrix} P_0(x) \\ P_1(x) \\ \vdots \end{pmatrix}, \quad \mathbf{f} := \begin{pmatrix} f_0 \\ f_1 \\ \vdots \end{pmatrix}.$$

This works analogously for bivariate orthogonal polynomials

$$f(x,y) = \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} P_{n,k}(x,y) f_{n,k}.$$

Jacobi operators

We can compute $xf(x)$ if $f(x)$ is given in coefficient vector form:

$$\mathbf{P}(x)^{\mathsf{T}}\mathrm{J}^{\mathsf{T}}\mathbf{f} = xf(x).$$

This is efficiently possible because the Jacobi polynomials satisfy a three term recurrence relationship, making J a tridiagonal operator.

Analogously on the triangle

We use the Jacobi polynomials shifted to the $[0, 1]$ interval and denote them by $\tilde{\mathbf{P}}^{(\alpha,\beta)}$, which allows us to write the bivariate Jacobi polynomials on the triangle as:

$$P_{k,n}^{(\alpha,\beta,\gamma)}(x,y) = (1-x)^k \tilde{P}_{n-k}^{(2k+\beta+\gamma+1,\alpha)}(x)\, \tilde{P}_k^{(\gamma,\beta)}\left(\frac{y}{1-x}\right).$$

As in the 1-dimensional case we can define Jacobi operators (now block tridiagonal) J_x and J_y, one for each variable:

$$\mathbf{P}(x,y)^{\mathsf{T}}\mathrm{J}_x^{\mathsf{T}}\mathbf{f}_\triangle = xf(x,y),$$

$$\mathbf{P}(x,y)^{\mathsf{T}}\mathrm{J}_y^{\mathsf{T}}\mathbf{f}_\triangle = yf(x,y).$$

Analogously on the triangle

We use the Jacobi polynomials shifted to the $[0,1]$ interval and denote them by $\tilde{\mathbf{P}}^{(\alpha,\beta)}$, which allows us to write the bivariate Jacobi polynomials on the triangle as:

$$P_{k,n}^{(\alpha,\beta,\gamma)}(x,y) = (1-x)^k \tilde{P}_{n-k}^{(2k+\beta+\gamma+1,\alpha)}(x)\, \tilde{P}_k^{(\gamma,\beta)}\left(\frac{y}{1-x}\right).$$

As in the 1-dimensional case we can define Jacobi operators (now block tridiagonal) J_x and J_y, one for each variable:

$$\mathbf{P}(x,y)^\mathsf{T} \mathrm{J}_x^\mathsf{T} \mathbf{f}_\triangle = xf(x,y),$$

$$\mathbf{P}(x,y)^\mathsf{T} \mathrm{J}_y^\mathsf{T} \mathbf{f}_\triangle = yf(x,y).$$

Spectral method for Volterra integral equations

The Volterra operator on coefficient space

We build the operator

$$\int_0^{1-x} f(y)\mathrm{d}y = \mathbf{P}(x)^\top \mathrm{W_Q} \mathrm{Q}_y \mathrm{E}_y \mathbf{f}_{[0,1]}$$

from two parts:

1. Q_y is the integral operator

$$\mathbf{P}(x)^\top \mathrm{W_Q} \mathrm{Q}_y \mathbf{f}_\triangle = \int_{y=0}^{1-x} f(x,y)\mathrm{d}y,$$

2. E_y extends a one-dimensional function on $[0,1]$ to the triangle:

$$\mathbf{P}(x,y)^\top \mathrm{E}_y \mathbf{f}_{[0,1]} = \mathbf{P}(x)^\top \mathbf{f}_{[0,1]}.$$

The Volterra operator on coefficient space

We build the operator

$$\int_0^{1-x} f(y)\mathrm{d}y = \mathbf{P}(x)^\mathsf{T} \mathrm{W}_\mathrm{Q} \mathrm{Q}_y \mathrm{E}_y \mathbf{f}_{[0,1]}$$

from two parts:

1. Q_y is the integral operator

$$\mathbf{P}(x)^\mathsf{T} \mathrm{W}_\mathrm{Q} \mathrm{Q}_y \mathbf{f}_\triangle = \int_{y=0}^{1-x} f(x,y)\mathrm{d}y,$$

2. E_y extends a one-dimensional function on $[0,1]$ to the triangle:

$$\mathbf{P}(x,y)^\mathsf{T} \mathrm{E}_y \mathbf{f}_{[0,1]} = \mathbf{P}(x)^\mathsf{T} \mathbf{f}_{[0,1]}.$$

The Volterra operator on coefficient space

We build the operator

$$\int_0^{1-x} f(y)\mathrm{d}y = \mathbf{P}(x)^\mathsf{T} \mathrm{W}_\mathrm{Q} \mathrm{Q}_y \mathrm{E}_y \mathbf{f}_{[0,1]}$$

from two parts:

1. Q_y is the integral operator

$$\mathbf{P}(x)^\mathsf{T} \mathrm{W}_\mathrm{Q} \mathrm{Q}_y \mathbf{f}_\triangle = \int_{y=0}^{1-x} f(x,y)\mathrm{d}y,$$

2. E_y extends a one-dimensional function on $[0,1]$ to the triangle:

$$\mathbf{P}(x,y)^\mathsf{T} \mathrm{E}_y \mathbf{f}_{[0,1]} = \mathbf{P}(x)^\mathsf{T} \mathbf{f}_{[0,1]}.$$

The operators

Using properties of the Jacobi polynomials one can derive

$$Q_y = \begin{pmatrix} \boxed{1} & & & & \\ & \boxed{1 \quad 0} & & & \\ & & \boxed{\ddots \quad \ddots \quad \ddots} \end{pmatrix}, \quad E_y = \begin{pmatrix} \boxed{\times} & & \\ & \boxed{\begin{matrix} \times \\ \times \end{matrix}} & \\ & & \boxed{\begin{matrix} \ddots \\ \ddots \\ \ddots \end{matrix}} \end{pmatrix}$$

where $E_y(n)_j = \frac{(-1)^{j+n}(2j-1)}{n}$. This also means that

$$(Q_y E_y)(n) = D_y(n) = \frac{(-1)^{n+1}}{n}.$$

Dealing with kernels

Assuming a monomial expansion for the kernel[1]

$$K(x,y) = \sum_{n=0}^{\infty} \sum_{j=0}^{n} k_{nj} x^{n-j} y^{j},$$

the primary part of the Volterra integration operator is

$$\begin{aligned} \mathrm{Q}_y K(\mathrm{J}_x^\top, \mathrm{J}_y^\top)\mathrm{E}_y &= \mathrm{Q}_y \left(\sum_{n=0}^{\infty} \sum_{j=0}^{n} k_{nj} (\mathrm{J}_x^\top)^{n-j} (\mathrm{J}_y^\top)^{j} \right) \mathrm{E}_y \\ &= \sum_{n=0}^{\infty} \sum_{j=0}^{n} k_{nj} (\mathrm{J}^\top)^{n-j} \mathrm{Q}_y \mathrm{E}_y (\mathrm{J}^\top)^{j}. \end{aligned}$$

where we made use of

$$\begin{aligned} \mathrm{Q}_y \mathrm{J}_x^\top \mathbf{f}_\triangle &= \mathrm{J}^\top \mathrm{Q}_y \mathbf{f}_\triangle, \\ \mathrm{J}_y^\top \mathrm{E}_y \mathbf{f}_{[0,1]} &= \mathrm{E}_y \mathrm{J}^\top \mathbf{f}_{[0,1]}. \end{aligned}$$

[1]We actually use a modified variation of Clenshaw's algorithm on the triangle due to see S. Olver, A. Townsend and G. Vasil (2019).

Dealing with kernels

Assuming a monomial expansion for the kernel[1]

$$K(x,y) = \sum_{n=0}^{\infty} \sum_{j=0}^{n} k_{nj} x^{n-j} y^j,$$

the primary part of the Volterra integration operator is

$$\mathrm{Q}_y K(\mathrm{J}_x^\top, \mathrm{J}_y^\top)\mathrm{E}_y = \mathrm{Q}_y \left(\sum_{n=0}^{\infty} \sum_{j=0}^{n} k_{nj} (\mathrm{J}_x^\top)^{n-j} (\mathrm{J}_y^\top)^j \right) \mathrm{E}_y$$

$$= \sum_{n=0}^{\infty} \sum_{j=0}^{n} k_{nj} (\mathrm{J}^\top)^{n-j} \mathrm{Q}_y \mathrm{E}_y (\mathrm{J}^\top)^j.$$

where we made use of

$$\mathrm{Q}_y \mathrm{J}_x^\top \mathbf{f}_\triangle = \mathrm{J}^\top \mathrm{Q}_y \mathbf{f}_\triangle,$$

$$\mathrm{J}_y^\top \mathrm{E}_y \mathbf{f}_{[0,1]} = \mathrm{E}_y \mathrm{J}^\top \mathbf{f}_{[0,1]}.$$

[1] We actually use a modified variation of Clenshaw's algorithm on the triangle due to see S. Olver, A. Townsend and G. Vasil (2019).

Dealing with kernels

Assuming a monomial expansion for the kernel[1]

$$K(x,y) = \sum_{n=0}^{\infty}\sum_{j=0}^{n} k_{nj} x^{n-j} y^{j},$$

the primary part of the Volterra integration operator is

$$\begin{aligned}
\mathrm{Q}_y K(\mathrm{J}_x^{\top}, \mathrm{J}_y^{\top})\mathrm{E}_y &= \mathrm{Q}_y \left(\sum_{n=0}^{\infty}\sum_{j=0}^{n} k_{nj} (\mathrm{J}_x^{\top})^{n-j} (\mathrm{J}_y^{\top})^{j} \right) \mathrm{E}_y \\
&= \sum_{n=0}^{\infty}\sum_{j=0}^{n} k_{nj} (\mathrm{J}^{\top})^{n-j} \mathrm{Q}_y \mathrm{E}_y (\mathrm{J}^{\top})^{j}.
\end{aligned}$$

where we made use of

$$\mathrm{Q}_y \mathrm{J}_x^{\top} \mathbf{f}_{\triangle} = \mathrm{J}^{\top} \mathrm{Q}_y \mathbf{f}_{\triangle},$$

$$\mathrm{J}_y^{\top} \mathrm{E}_y \mathbf{f}_{[0,1]} = \mathrm{E}_y \mathrm{J}^{\top} \mathbf{f}_{[0,1]}.$$

[1] We actually use a modified variation of Clenshaw's algorithm on the triangle due to see S. Olver, A. Townsend and G. Vasil (2019).

The method for integral equations

Equations of first kind, $\mathrm{V}_K u = g$, turn into

$$\tilde{\mathbf{P}}^{(1,0)}(x)^\mathsf{T} \mathrm{Q}_y K(\mathrm{J}_x^\mathsf{T}, \mathrm{J}_y^\mathsf{T}) \mathrm{E}_y \mathbf{u} = \tilde{\mathbf{P}}^{(1,0)}(x)^\mathsf{T} \mathbf{q},$$

where now $\mathbf{q}$ is the coefficient vector of $q(x) = \frac{g(x)}{1-x}$.

Equations of second kind, $(I + \mathrm{V}_K)u = g$, turn into

$$\tilde{\mathbf{P}}^{(1,0)}(x)^\mathsf{T} \left(\mathbb{1} - (\mathbb{1} - \mathrm{J}^\mathsf{T}) \mathrm{Q}_y K(\mathrm{J}_x^\mathsf{T}, \mathrm{J}_y^\mathsf{T}) \mathrm{E}_y\right) \mathbf{u} = \tilde{\mathbf{P}}^{(1,0)}(x)^\mathsf{T} \mathbf{g}.$$

Quick notes on convergence

Sketch for second kind

$$\begin{tikzcd}
L^2(0,1) \arrow[r, "\mathcal{V}_K"] \arrow[d, "\mathcal{E}"'] & L^2(0,1) \\
\ell^2 \arrow[r, "\mathrm{V}_K"'] & \ell^2 \arrow[u, "\mathcal{E}^{-1}"']
\end{tikzcd}$$

For Volterra integral equations of second kind the operator to be inverted is of the form $(\mathbb{1} + \mathrm{V}_K)$ and V_K compact.

Sketch for first kind (I)

Since V_K compact from ℓ^2 to ℓ^2 we instead consider $\mathrm{V}_K : \ell^2 \to \ell^2_1$ where ℓ^2_λ denotes the Banach space with norm

$$\|\mathbf{u}\|_{\ell^2_\lambda} = \sqrt{\sum_{n=0}^{\infty} \left((1+n)^\lambda |u_n|\right)^2} < \infty.$$

Then the operator can be brought into the form

$$\mathrm{V}_K = \mathrm{D}(\mathrm{T}_f + \mathrm{C}),$$

where T is a symmetric Toeplitz operator with real entries and with symbol f, C is compact and D is diagonal, bounded and invertible.

Sketch for first kind (I)

Since V_K compact from ℓ^2 to ℓ^2 we instead consider $\mathrm{V}_K : \ell^2 \to \ell^2_1$ where ℓ^2_λ denotes the Banach space with norm

$$\|\mathbf{u}\|_{\ell^2_\lambda} = \sqrt{\sum_{n=0}^{\infty} \left((1+n)^\lambda |u_n|\right)^2} < \infty.$$

Then the operator can be brought into the form

$$\mathrm{V}_K = \mathrm{D}(\mathrm{T}_f + \mathrm{C}),$$

where T is a symmetric Toeplitz operator with real entries and with symbol f, C is compact and D is diagonal, bounded and invertible.

Sketch for first kind (II)

The symbol of the Toeplitz operator part is uniquely determined by the coefficients of the kernel to be

$$f(z) = \sum_{n=0}^{M} \sum_{j=0}^{n} k_{nj} \cos^{2n}\left(\frac{\theta}{2}\right) \qquad \text{where} \qquad z = \mathrm{e}^{\mathrm{i}\theta}.$$

The resulting condition for convergence of the method is found to be:

$$\forall x \in [0,1] : K(x,x) \neq 0.$$

Implementation in Julia under ApproxFun.jl framework

Three examples

We seek numerical solutions u_1, u_2 and u_3 to the following three Volterra integral equations of second kind.

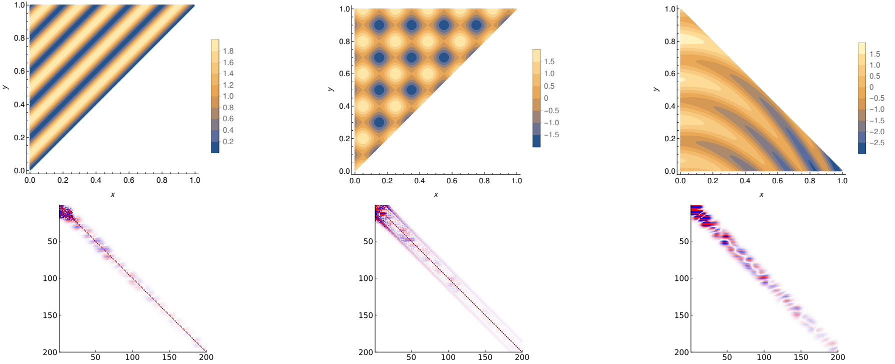
Let $C(x) = \frac{e^{-10\pi x}(1+20\pi)-2+\cos(10\pi x)+\sin(10\pi x)}{20\pi}$.

$$u_1(x) = C(x) + \int_0^x \left(1 - \cos\left(10\pi x - 10\pi y\right)\right) u_1(y)\mathrm{d}y \tag{1}$$

$$u_2(x) = \frac{e^{\frac{x}{2}}}{\pi} + \int_0^x \left(\sin(10\pi x) + \cos(10\pi y)\right) u_2(y)\mathrm{d}y \tag{2}$$

$$u_3(x) = e^{x^2-2x} + \int_0^{1-x} \left(-2x + y + \sin(25x^2 + 8\pi y)\right) u_3(y)\mathrm{d}y \tag{3}$$

A look at the kernels and operators

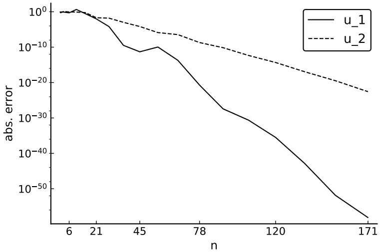


(a) $K_1(x, y)$ (b) $K_2(x, y)$ (c) $K_3(x, y)$

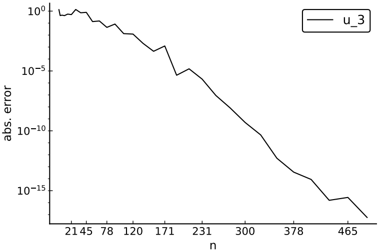
Figure: Kernel contour plots and operator spy plots.

Convergence of numerical experiments

Implementation in Julia under ApproxFun.jl framework



(a)



(b)

Figure: Absolute errors for equations (1–3). $u_1(x)$ is compared to the analytic solution, $u_2(x)$ and $u_3(x)$ are compared to a solution computed with $n = 5050$.

Discussion

- The Volterra integral operator is banded on an appropriate basis of orthogonal polynomials.
- This can be used in a highly efficient sparse spectral method for Volterra integrals and integral equations.
- The method is not restricted to convolution kernels.
- We have a working implementation of this method under ApproxFun.jl framework.
- As it is the method only works for linear Volterra integral equations but an extension to non-linear cases is conceivable - we are working on it.

Discussion

- The Volterra integral operator is banded on an appropriate basis of orthogonal polynomials.
- This can be used in a highly efficient sparse spectral method for Volterra integrals and integral equations.
- The method is not restricted to convolution kernels.
- We have a working implementation of this method under ApproxFun.jl framework.
- As it is the method only works for linear Volterra integral equations but an extension to non-linear cases is conceivable - we are working on it.

Discussion

- The Volterra integral operator is banded on an appropriate basis of orthogonal polynomials.
- This can be used in a highly efficient sparse spectral method for Volterra integrals and integral equations.
- The method is not restricted to convolution kernels.
- We have a working implementation of this method under ApproxFun.jl framework.
- As it is the method only works for linear Volterra integral equations but an extension to non-linear cases is conceivable - we are working on it.

Discussion

- The Volterra integral operator is banded on an appropriate basis of orthogonal polynomials.
- This can be used in a highly efficient sparse spectral method for Volterra integrals and integral equations.
- The method is not restricted to convolution kernels.
- We have a working implementation of this method under ApproxFun.jl framework.
- As it is the method only works for linear Volterra integral equations but an extension to non-linear cases is conceivable - we are working on it.

Discussion

- The Volterra integral operator is banded on an appropriate basis of orthogonal polynomials.
- This can be used in a highly efficient sparse spectral method for Volterra integrals and integral equations.
- The method is not restricted to convolution kernels.
- We have a working implementation of this method under ApproxFun.jl framework.
- As it is the method only works for linear Volterra integral equations but an extension to non-linear cases is conceivable - we are working on it.

Clenshaw's algorithm

The algorithm makes use of the polynomial basis' recurrence relationships to reduce function evaluation to the solution of a triangular linear system:

$$\mathrm{Q}_y K(\mathrm{J}_x, \mathrm{J}_y)\mathrm{E}_y = (\mathbf{e}_0 \otimes \mathbb{1})\mathcal{L}_V^{-\mathsf{T}}(\mathbf{K}_\triangle \otimes \mathrm{Q}_y \mathrm{E}_y),$$

with

$$\mathcal{L}_V = \begin{pmatrix} (\mathbb{1}_1 \otimes \mathbb{1}) & & & \\ (A_0^x \otimes \mathbb{1}) - (\mathbb{1}_1 \otimes \mathrm{J}\diamond) & (B_0^x \otimes \mathbb{1}) & & \\ (A_0^y \otimes \mathbb{1}) - (\mathbb{1}_1 \otimes \diamond\mathrm{J}) & (B_0^y \otimes \mathbb{1}) & & \\ (C_0^x \otimes \mathbb{1}) & (A_1^x \otimes \mathbb{1}) - (\mathbb{1}_2 \otimes \mathrm{J}\diamond) & (B_1^x \otimes \mathbb{1}) & \\ (C_0^y \otimes \mathbb{1}) & (A_1^x \otimes \mathbb{1}) - (\mathbb{1}_2 \otimes \diamond\mathrm{J}) & (B_1^y \otimes \mathbb{1}) & \\ & \ddots & \ddots & \ddots \end{pmatrix}.$$

After appropriate preconditioning this system can be solved via backward substitution, see S. Olver, A. Townsend and G. Vasil (2019).