Oblivious Convolution Quadrature

Andrey Petukhov

ETH Zurich

December 7, 2009

Outline

The problem

Effective approximation of a continuous convolution

$$\int_0^t f(t-\tau)g(\tau)\,\mathrm{d}\tau$$

at discrete time steps $t = h, 2h, ..., Nh = T$.

Effective $\Longrightarrow$ fast and memory-saving.

Input data:

- the Laplace transform $F(s)$ rather than $f(t)$ is known analytically;
- $g(t)$ (given explicitly or implicitly).

General assumptions

We assume a *sectorial* Laplace transform $F(s) = \mathcal{L}f(t)$:

- $F(s)$ is analytic in a sector $|\arg(s-\sigma)| < \pi - \varphi$ with $0 < \varphi < \frac{1}{2}\pi$
- in this sector $|F(s)| \leq M|s|^{-\nu}$ for some $M > 0$ and $\nu > 0$

The inverse Laplace transform is then given by

$$f(t) = \frac{1}{2\pi i} \int_{\Gamma} e^{t\lambda} F(\lambda)\, \mathrm{d}\lambda, \qquad t > 0.$$

The function $f(t)$ is thus analytic in $t > 0$, satisfies $|f(t)| \leq Ct^{\nu-1}e^{ct}$ and is therefore locally integrable.

Convolution quadrature

Implementation of a quadrature formula with a fixed step size $h > 0$

$$\int_0^t f(t-\tau)g(\tau)\,\mathrm{d}\tau \approx \sum_{j=0}^{n} w_{n-j} g(jh), \qquad n = 1, \dots, N.$$

The convolution weights are the coefficients of the power series (obtained from linear multistep methods):

$$\sum_{n=0}^{\infty} w_n \zeta^n = F\left(\frac{\delta(\zeta)}{h}\right).$$

Complexity of different algorithms

Direct computation of convolution quadrature:

- $O(N^2)$ multiplications;
- $O(N)$ evaluations of Laplace transform $F(s)$ for computing the weights;
- $O(N)$ active memory used.

Oblivious convolution quadrature:

- $O(N \log N)$ multiplications;
- $O(\log N)$ evaluations of Laplace transform $F(s)$;
- $O(\log N)$ active memory used.

Approximation of inverse Laplace transform

Apply trapezoidal rule to a parametrization of the contour integral:

$$f(t) = \frac{1}{2\pi i} \int_{\Gamma} e^{t\lambda} F(\lambda)\,\mathrm{d}\lambda \approx \sum_{j=-K}^{K} w_{n-j} F(\lambda_j) e^{t\lambda_j}$$

Idea: use local approximation on fast growing time intervals I_l covering $[\Delta t, T]$:

$$I_l = [B^{l-1}\Delta t, (2B^l - 1)\Delta t],$$

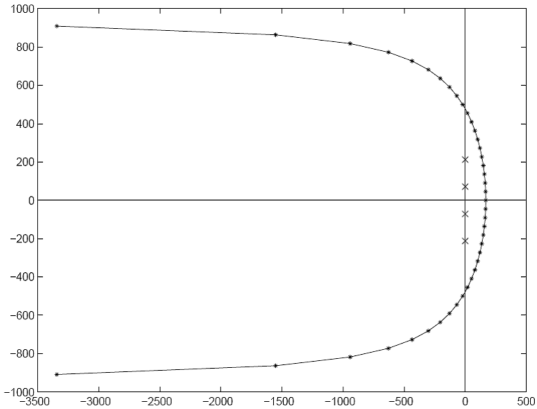
with *base* $B > 1$ and l - integer numbers.

Use different contours Γ_l for different numbers l.

Choice of the contour (I)

Talbot contour:

$$\theta \in (-\pi, \pi) \to \Gamma_l \quad \theta \mapsto \gamma_l(\theta) = \sigma + \mu_l(\theta \cot \theta + i\nu\theta)$$



Choice of the contour (II)

Weights and quadrature points on Talbot contour:

$$w_j^{(l)} = -\frac{j}{2(K+1)}\gamma_l'(\theta_j), \quad \lambda_j^{(l)} = \gamma_l(\theta_j), \quad \theta_j = \frac{j\pi}{K+1}$$

A trapezoidal rule with $2K+1$ nodes is used in the parameter domain.

Error estimates: exponential convergence

$$\frac{|f(t) - \widetilde{f}(t)|}{|f(t)|} = O(\exp(-ct\sqrt{K}))$$

Reduction to ordinary differential equations (I)

For general boundary points $a < b$:

$$\int_a^b f(t-\tau)g(\tau)\,\mathrm{d}\tau = \int_a^b \frac{1}{2\pi i}\int_\Gamma F(\lambda)e^{(t-\tau)\lambda}\mathrm{d}\lambda g(\tau)\mathrm{d}\tau =$$

$$= \frac{1}{2\pi i}\int_\Gamma F(\lambda)e^{(t-b)\lambda}\int_a^b e^{(b-\tau)\lambda}g(\tau)\mathrm{d}\tau\mathrm{d}\lambda.$$

The integral $y(b,a,\lambda) = \int_a^b e^{(b-\tau)\lambda}g(\tau)\mathrm{d}\tau$ is the solution at time b for the initial value problem

$$y' = \lambda y + g, \qquad y(a) = 0.$$

Reduction to ordinary differential equations (II)

If $[t-b, t-a] \subset I_l \Longrightarrow$ integral over the Talbot contour $\Gamma = \Gamma_l \Longrightarrow$ trapezoidal approximation:

$$\int_a^b f(t-\tau)g(\tau)\,\mathrm{d}\tau \approx \int_a^b \sum_{j=-K}^{K} w_j F(\lambda_j) e^{(t-\tau)\lambda_j} g(\tau)\,\mathrm{d}\tau$$

$$= \sum_{j=-K}^{K} w_j F(\lambda_j) e^{(t-b)\lambda_j} y(b, a, \lambda_j)$$

For all $y(b, a, \lambda_j)$ we get $2K+1$ differential equations.

Approximate solution for obtained ODEs

- Piecewise linear approximation of $g(t)$.
- After this solving exactly

Set $g_n = g(a + n\Delta t) \Longrightarrow$ recursively obtain $y_n \approx y(a + n\Delta t)$:

$$y_{n+1} = e^{\Delta t\lambda} y_n + h \int_0^1 e^{(1-\theta)\Delta t\lambda}(\theta g_{n+1} + (1-\theta) g_n)\,\mathrm{d}\theta$$

$$= y_n + \frac{e^{\Delta t\lambda} - 1}{\Delta t\lambda}(\Delta t\lambda y_n + \Delta t g_n + \Delta t \frac{g_{n+1} - g_n}{\Delta t\lambda}) - \Delta t \frac{g_{n+1} - g_n}{\Delta t\lambda}.$$

Basic ideas

General approximation of the convolution:

$$\int_a^b f(t-\tau)g(\tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j F(\lambda_j) e^{(t-b)\lambda_j} y(b,a,\lambda_j)$$

We have:

- Algorithm to compute approximate inversion of Laplace transform by trapezoidal rule on a Talbot contour Γ_l for the intervals $I_l = [B^{l-1}\Delta t, (2B^l - 1)\Delta t]$;
- Algorithm to compute $y^{(l)}(b, a, \lambda_j)$ at the same intervals.

Step 1

Compute the convolution integral at $t = \Delta t$ by approximating $g(\tau)$ linearly:

$$\int_0^{\Delta t} f(\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \int_0^{\Delta t} f(\Delta t - \tau) g(0)\,\mathrm{d}\tau + \int_0^{\Delta t} f(\Delta t - \tau)\tau\,\mathrm{d}\tau \frac{g(\Delta t) - g(0)}{\Delta t}$$

Approximate the integrals as the inverse Laplace transforms of $F(s)/s$ and $F(s)/s^2$:

$$\phi_1 = \int_0^{\Delta t} f(\Delta t - \tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j F(\lambda_j)/\lambda_j e^{\Delta t \lambda_j}$$

$$\phi_2 = \int_0^{\Delta t} f(\Delta t - \tau)\tau\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j F(\lambda_j)/\lambda_j^2 e^{\Delta t \lambda_j}$$

Step 1 (remarks)

By linear change of variable $\tau \to \tau + k\Delta t$ we get:

$$\int_{k\Delta t}^{(k+1)\Delta t} f(k\Delta t - \tau)\,\mathrm{d}\tau = \int_{0}^{\Delta t} f(\Delta t - \tau)\,\mathrm{d}\tau = \phi_1$$

$$\int_{k\Delta t}^{(k+1)\Delta t} f(k\Delta t - \tau)\tau\,\mathrm{d}\tau = \int_{0}^{\Delta t} f(\Delta t - \tau)\tau\,\mathrm{d}\tau = \phi_2$$

ϕ_1 and ϕ_2 can be *precomputed* with high accuracy.

The first step - "Near Past".

Step 2

Convolution integral at $t = 2\Delta t \Longrightarrow$ splitting integrals in the integrals over $[0, \Delta t]$ and $[\Delta t, 2\Delta t]$:

$$\int_{\Delta t}^{2\Delta t} f(2\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \phi_1 g(\Delta t) + \phi_2 \frac{g(2\Delta t) - g(\Delta t)}{\Delta t}$$

$$\int_{0}^{\Delta t} f(2\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j^{(1)} F(\lambda_j^{(1)}) e^{\Delta t \lambda_j^{(1)}} y(\Delta t, 0, \lambda_j^{(1)})$$

(for Talbot contour Γ_1 corresponding to $I_1 = [\Delta t, 3\Delta t]$, solutions of $2K + 1$ ODEs required).

Step 3

Convolution integral at $t = 3\Delta t \Longrightarrow$ splitting integrals in the integrals over $[0, 2\Delta t]$ and $[2\Delta t, 3\Delta t]$:

$$\int_{2\Delta t}^{3\Delta t} f(3\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \phi_1 g(2\Delta t) + \phi_2 \frac{g(3\Delta t) - g(2\Delta t)}{\Delta t}$$

$$\int_{0}^{2\Delta t} f(3\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j^{(1)} F(\lambda_j^{(1)}) e^{\Delta t \lambda_j^{(1)}} y(2\Delta t, 0, \lambda_j^{(1)})$$

(advancing the solutions of ODEs for Γ_1 from Δt to $2\Delta t$ required).

Step 4 (I)

Convolution integral at $t = 4\Delta t \Longrightarrow$ attempt to split integrals :

$$\int_{3\Delta t}^{4\Delta t} f(4\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \phi_1 g(3\Delta t) + \phi_2 \frac{g(4\Delta t) - g(3\Delta t)}{\Delta t}$$

Approximating the integral over $[0, 3\Delta t] \Longrightarrow$ we can't use the contour Γ_1 for the whole integral as $t - \tau \in [\Delta t, 4\Delta t] \notin I_1$.

Idea: split the integral over $[0, 3\Delta t]$ in two integrals over $[0, 2\Delta t]$ and $[2\Delta t, 3\Delta t]$.

Step 4 (II)

The integral over $[2\Delta t, 3\Delta t]$ with $t - \tau \in I_1$:

$$\int_{2\Delta t}^{3\Delta t} f(4\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j^{(1)} F(\lambda_j^{(1)}) e^{\Delta t \lambda_j^{(1)}} y(3\Delta t, 2\Delta t, \lambda_j^{(1)})$$

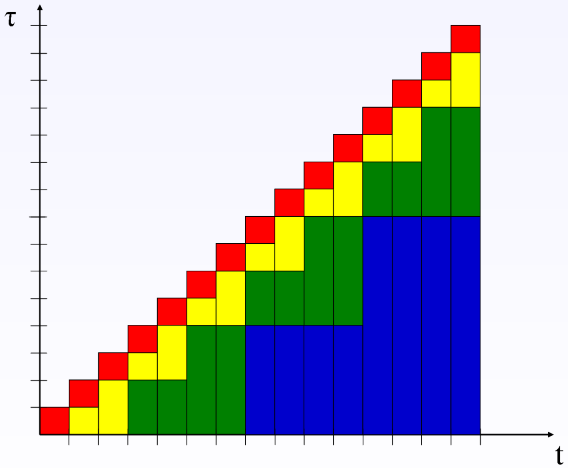
(approximated on Γ_1).

The integral over $[0, 2\Delta t]$ with $t - \tau \in [2\Delta t, 4\Delta t] \subset I_2 = [2\Delta t, 7\Delta t]$:

$$\int_{0}^{2\Delta t} f(4\Delta t - \tau) g(\tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j^{(2)} F(\lambda_j^{(2)}) e^{2\Delta t \lambda_j^{(2)}} y(2\Delta t, 0, \lambda_j^{(2)})$$

(approximated on Γ_2, solutions of another $2K + 1$ ODEs required).

Decomposition for the base $B = 2$



$\mathbf{t} - \tau < \mathbf{\Delta t}$ $\quad$ $\mathbf{t} - \tau \in \mathbf{I_1}$ $\quad$ $\mathbf{t} - \tau \in \mathbf{I_2}$ $\quad$ $\mathbf{t} - \tau \in \mathbf{I_3}$

Organization of the algorithm

- All the ODEs for all contours Γ_l are advanced by one time step in every step $t \to t + \Delta t$ of algorithm.
- Past values of $g(t)$ need NOT be kept in memory.
- Only present values of $g(t)$ and possibly one past value of $y(t, 0, \lambda_j^{(l)})$ for all j and all l must be stored.

Complexity of the algorithm

The *number of operations* is proportional to NLK:

- N - the number of time steps;
- $2K + 1$ - the number of quadrature points on each contour;
- $L \leq \log_2 N$ - the number of different contours.

Memory required is approximately $5LN$:

- storing quadrature points $\lambda_j^{(l)}$;
- storing exponentials $\exp \Delta t \lambda_j^{(l)}$;
- storing the weighted functions $w_j^{(l)} F(\lambda_j^{(l)})$.

The algorithm is highly *parallelizable*.

Ideas on error estimates

Recall the main approximation formula:

$$\int_a^b f(t-\tau)g(\tau)\,\mathrm{d}\tau \approx \sum_{j=-K}^{K} w_j F(\lambda_j) e^{(t-b)\lambda_j} y(b,a,\lambda_j)$$

We totally approximate:

$$\int_a^b f(t-\tau)g(\tau)\,\mathrm{d}\tau \approx \int_a^b \widetilde{f}(t-\tau)\widetilde{g}(\tau)\,\mathrm{d}\tau$$

Two sources of error:

- approximation on inverse Laplace transform (exponential convergence);
- linear approximation of $g(\tau)$:
  $||\widetilde{g}(\tau) - g(\tau)|| \leq \frac{1}{8} h^2 \max_{t_{n-1} \leq t \leq t_n} ||g''(t)||$

Ideas on improvements

Improvements in solving ODEs:

- approximation of $g(\tau)$ with higher-order interpolants;
- applying numerical methods with high order of accuracy for solving ODEs (Runge-Kutta methods).

Ideas on contour integrals:

- hyperbolas can be used instead of Talbot contours (higher accuracy $\Longrightarrow$ larger intervals can be chosen$\Longrightarrow$ complexity is reduced).
  Nevertheless Talbout contours are less sensitive to the choice of the parameters and the Laplace transform of the functions.

References

1. C. Lubich and A. Schädle, *Fast convolution for non-reflecting boundary conditions*, SIAM J. Sci. Comp., 24 (2002), pp. 161–182.
2. R. Hiptmair and A. Schädle, *Non-reflecting boundary conditions for Maxwell's equations*, Computing, 71 (2003), pp. 165–292.
3. A. Schädle, M. Lopez-Fernandez and C. Lubich, *Fast and oblivious convolution quadrature*, SIAM J. Sci. Comp., 28(2)(2003) pp. 421-438.