FAST MULTISCALE DIFFUSION ON GRAPHS

S. Marcotte[1], A. Barbe[2], R. Gribonval[2], T. Vayer[2], M. Sebban[3], P. Borgnat[4], P. Gonçalves[2]

[1]ENS Rennes [2]Univ de Lyon, Inria, CNRS, ENSL, LIP
[3]Univ Lyon, UJM-St-Etienne, CNRS, Lab. Hubert Curien [4]Univ Lyon, CNRS, ENSL, Lab. de Physique

ABSTRACT

Diffusing a graph signal at multiple scales requires to compute the action of the exponential of as many versions of the Laplacian matrix. Considering the truncated Chebyshev polynomial approximation of the exponential, we derive a tightened bound on the approximation error, allowing thus for a better estimate of the polynomial degree that reaches a prescribed error. We leverage the properties of these approximations to factorize the computation of the action of the diffusion operator over multiple scales, thus drastically reducing its computational cost.

***Index Terms*—** Approximate computing, Chebyshev approximation, Computational efficiency, Estimation error.

1. INTRODUCTION

The matrix exponential operator has applications in numerous domains, ranging from time integration of ordinary differential equations [1] or network analysis [2] to various simulation problems (like power grids [3] or nuclear reactions [4]) or machine learning [5]. In graph signal processing, it appears in the diffusion process of a graph signal – an analog on graphs of low-pass filtering.

Given a graph $\mathcal{G}$ and its combinatorial Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, with $\mathbf{A}$ the adjacency matrix and $\mathbf{D}$ the diagonal degree matrix, let x be a signal on this graph (a vector containing a value at each node), the *diffusion of x in $\mathcal{G}$* is defined by the equation $\frac{dw}{d\tau} = -\mathbf{L}w$ with $w(0) = x$ [6]. It admits a closed-form solution $w(\tau) = \exp(-\tau\mathbf{L})x$ involving the *heat kernel* $\tau \to \exp(-\tau\mathbf{L})$, which features the matrix exponential.

Applying the exponential of a matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ to a vector $x \in \mathbb{R}^n$ can be achieved by computing the matrix $\mathbf{B} = \exp(\mathbf{M})$ followed by the matrix-vector product $\mathbf{B}x$. However, the computational cost of these operations, quadratic in n, become quickly computationally prohibitive in high dimension. On the other hand, multiscale graph representations such as graph wavelets [7], graph-based machine learning methods [5], rely on graph diffusion at different scales that imply applications of the matrix exponential for various multiples of the graph Laplacian.

To speedup such repeated computations one can use a well-known technique based on approximations of the (scalar) exponential function using Chebyshev polynomials. We build on the fact that polynomial approximations [8] can significantly reduce the computational burden of approximating $\exp(\mathbf{M})x$ with good precision when $\mathbf{M} = -\tau\mathbf{L}$ and $\mathbf{L}$ is symmetric positive semi-definite (SPSD) and sparse. This is notably the case when $\mathbf{L}$ is the Laplacian of an undirected graph. The principle is to approximate the exponential as a low-degree polynomial in $\mathbf{M}$ and compute its action, $\exp(\mathbf{M})x \approx p(\mathbf{M})x := \sum_{k=0}^{K} a_k\mathbf{M}^k\, x$. Several methods exist, some requiring the explicit computation of coefficients associated with a particular choice of polynomial basis, others, including Krylov-based techniques, not requiring explicit evaluation of the coefficients but relying on an iterative determination [9] of the polynomial approximation on the subspace spanned by $\left\{x, \mathbf{M}x, \cdots, \mathbf{M}^K x\right\}$.

Our contribution is twofold. First, we devise a new bound on the approximation error of truncated Chebyshev expansions of the exponential, that improves upon existing works [10, 11, 12]. This avoids unnecessary computations by determining a small truncation degree K to achieve a prescribed error. Second, we propose to compute $\exp(-\tau\mathbf{L})$ at different scales $\tau \in \mathbb{R}_+$ faster, by reusing the calculations of the action of Chebyshev polynomials on x and combining them with adapted coefficients for each scale τ. This is particularly efficient for multiscale problems with arbitrary values of τ, unlike [13] which is limited to linear spacing.

In Section 2 we recall Chebyshev polynomials approximation and derive new bounds on the coefficients (Cor. 2.2). Then, we approximate in Section 3 the matrix exponential with controlled complexity and error (Lemma 3.1) with our new error bounds. Section 4 ends with numerical validations.

2. CHEBYSHEV APPROXIMATION

The Chebyshev polynomials of the first kind are characterized by the identity $T_k(\cos(\theta)) = \cos(k\theta)$. They can be computed from $T_0(t) = 1$, $T_1(t) = t$ and the recurrence relation:

$$T_{k+2}(t) = 2tT_{k+1}(t) - T_k(t). \tag{1}$$

Work supported by the ACADEMICS grant of IDEXLYON, PIA operated by ANR-16-IDEX-0005, and supported in part by the AllegroAssai ANR project ANR-19-CHIA-0009.

The *Chebyshev series decomposition* of a function $f : [-1,1] \to \mathbb{R}$ takes on the form:

$$f(t) = \frac{c_0}{2} + \sum_{k\geq 1} c_k \cdot T_k(t), \tag{2}$$

and the *Chebyshev coefficients* have the following expression:

$$c_k = \frac{2}{\pi}\int_0^\pi \cos(k\theta)\, f(\cos(\theta))\,\mathrm{d}\theta. \tag{3}$$

By truncating the sum in the series (2) to some $K < +\infty$, we obtain an approximation of the function f.[1]

Chebyshev series of the exponential. Here, we focus on approximating the function $h_\tau : \lambda \in [0,2] \to \exp(-\tau\lambda)$, that will serve as a starting point to obtain a low-degree polynomial approximations of the matrix exponential $\exp(-\tau\mathbf{L})$ for symmetric positive semi-definite matrices whose largest eigenvalue satisfies $\lambda_{\max} = 2$ (see Section 3).

Using the change of variable: $t = (\lambda - 1) \in [-1,1]$ and the Chebyshev series of Eq. (2), we get:

$$h_\tau(\lambda) = \frac{1}{2}c_0(\tau) + \sum_{k=1}^{\infty} c_k(\tau)\tilde{T}_k(\lambda), \tag{4}$$

where, for any $k \in \mathbb{N}$, $\tilde{T}_k(\lambda) = T_k(\lambda - 1)$ and

$$c_k(\tau) = \frac{2}{\pi}\int_0^\pi \cos(k\theta)\exp(-\tau(\cos(\theta)+1))\,\mathrm{d}\theta. \tag{5}$$

Truncating the series (4) to K yields a polynomial approximation of h_τ of degree K whose quality can be controlled.

Chebyshev coefficients of the exponential operator. Numerical evaluation of the coefficients using the integral formulation of Eq. (3) is in general prohibitive. However, in the exponential case, the coefficients in Eq. (5) have a close form expression [15]:

$$c_k(\tau) = 2I_k(\tau)\cdot\exp(-\tau) = 2\cdot Ie_k(-\tau),$$

where $I_k(\cdot)$ is the modified Bessel function of the first kind and $Ie_k(\cdot)$ is the exponentially scaled modified Bessel function of the first kind. Moreover, the following lemma yields another expression for the coefficients c_k of Eq. (5), which will show more relevant to bound the error of the truncated Chebyshev expansion.

Lemma 2.1 ([14], Equation 2.91)**.** *Let f be a function expressed as an infinite power series $f(t) = \sum_{i=0}^{\infty} a_i t^i$ and assume that this series is uniformly convergent on $[-1,1]$. Then, the Chebyshev coefficients of f can be expressed as:*

$$c_k = \frac{1}{2^{k-1}}\sum_{i=0}^{\infty}\frac{1}{2^{2i}}\binom{k+2i}{i}a_{k+2i}. \tag{6}$$

[1]For theoretical aspects of the approximation by Chebyshev polynomials (and other polynomial basis) we refer the reader to [14].

Corollary 2.2. *For the particular function $\tilde{h}_\tau(t) := \exp(-\tau(t+1))$, $t\in[-1,1]$, the coefficients of its Chebyshev series read:*

$$c_k = (-1)^k d_k \bar{c}_k, \text{ with}$$
$$\bar{c}_k = 2\left(\tfrac{\tau}{2}\right)^k \tfrac{\exp(-\tau)}{k!} \;;\; d_k = \textstyle\sum_{i=0}^{\infty}\left(\tfrac{\tau}{2}\right)^{2i}\tfrac{k!}{i!(k+i)!},$$
$$\text{and in addition, } 1 \leq d_k \leq \min\left(\exp\left(\tfrac{(\tau/2)^2}{k+1}\right), \cosh(\tau)\right).$$

Proof. Posing $C = \tau/2$, we have the power series expansions:

$$\tilde{h}_\tau(t) = \exp(-2C)\exp(-2Ct) = \sum_{i=0}^{\infty}\exp(-2C)\frac{(-2C)^i}{i!}t^i,$$

and, using Lemma 2.1, for each $k \in \mathbb{N}$, we obtain:

$$c_k = (-1)^k C^k 2\exp(-2C)\sum_{i=0}^{\infty} C^{2i}\frac{1}{i!(k+i)!} = (-1)^k \bar{c}_k d_k.$$

For any integers k and i, $k!/(k+i)! \leq \min(1/i!, 1/(k+1)^i)$ and $1/(i!)^2 = \binom{2i}{i}/(2i)! \leq 2^{2i}/(2i)!$, which leads to:

$$d_k = \sum_{i=0}^{\infty}\frac{C^{2i}}{i!}\frac{k!}{(k+i)!} \leq \min\left(\sum_{i=0}^{\infty}\frac{C^{2i}}{i!}\frac{1}{(k+1)^i}, \sum_{i=0}^{\infty}\frac{C^{2i}}{i!i!}\right)$$
$$\leq \min\left(e^{\frac{C^2}{k+1}}, \sum_{i=0}^{\infty}\frac{C^{2i}2^{2i}}{(2i)!}\right) = \min\left(e^{\frac{C^2}{k+1}}, \cosh(2C)\right). \quad \square$$

3. APPROXIMATION OF THE MATRIX EXPONENTIAL

The extension of a univariate function $f : \mathbb{R} \to \mathbb{R}$ to symmetric matrices $\mathbf{L} \in \mathbb{R}^{n\times n}$ exploits the eigen-decomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{\Lambda} = \mathtt{diag}(\lambda_i)_{1\leq i\leq n}$, to define the action of f as $f(\mathbf{L}) := \mathbf{U}\mathtt{diag}(f(\lambda_i))\mathbf{U}^\top$. When $f(t) = t^k$ for some integer k, we have $f(\mathbf{L}) = \mathbf{L}^k$, and the definition matches with the intuition when f is polynomial or analytic.

While the function of a matrix can be computed by taking the same function applied to its eigenvalues, diagonalizing the matrix is computationally prohibitive. On the other hand, computing the matrix $f(\mathbf{L})$ itself is rarely required, as it is often its *action* on a given vector that is of interest. This remark enables faster methods, notably those using polynomial approximations. When the function f admits a Taylor expansion, a natural choice for the polynomial p is a truncated version of the Taylor series [13]. Other polynomial bases exist, such as the Padé polynomials or, in our case, the Chebyshev polynomials [11, 16, 17, 18], leading to approximation errors that decay exponentially with the polynomial degree K.

Chebyshev approximation of a matrix exponential. Consider $\mathbf{L}$ any SPSD matrix of largest eigenvalue $\lambda_{\max} = 2$ (adaptations to matrices whose largest eigenvalue is larger than 2 will be discussed in the experimental section). To approximate the action of $\exp(-\tau\mathbf{L})$, where $\tau \geq 0$, we use

the matrix polynomial $p_K(\mathbf{L})$ where $p_K(\lambda)$ is the polynomial obtained by truncating the series (4). The truncation degree K offers a compromise between computational speed and numerical accuracy. The recurrence relations (1) on Chebyshev polynomials extends to the calculus of $\tilde{T}_k(\mathbf{L})x = T_k(\mathbf{L} - \mathbf{Id})x$. Given a polynomial degree K, computing $p_K(\mathbf{L})x$ requires K matrix-vector products for the polynomials, and $K+1$ Bessel functions evaluation for the coefficients. This cost is dominated by the K matrix-vector products, which can be low if $\mathbf{L}$ is a sparse matrix.

Generic bounds on relative approximation errors. Denote p_K the polynomial obtained by truncating the Chebyshev expansion (4) at degree k. For a given input vector $x \neq 0$, we can measure the approximation error relative to $\| x \|_2$ as:

$$\epsilon_K(x) := \frac{\| \exp(-\tau\mathbf{L})x - p_K(\mathbf{L})x\|_2^2}{\|x\|_2^2}. \tag{7}$$

Expressing $\exp(-\tau\mathbf{L})$ and $p_K(\mathbf{L})$ in an orthonormal eigenbasis of $\mathbf{L}$ yields a worst-case relative error:

$$\epsilon_K := \sup_{x\neq 0} \epsilon_K(x) = \max_i |h_\tau(\lambda_i) - p_K(\lambda_i)|^2 \leq \|h_\tau - p_K\|_\infty^2$$

with $\lambda_i \in [0, \lambda_{\max}]$ the eigenvalues of $\mathbf{L}$ and $\|g\|_\infty := \sup_{\lambda\in[0,\lambda_{\max}]} |g(\lambda)|$.

Lemma 3.1. *Consider $\tau \geq 0$, h_τ as in Section 2, and $\mathbf{L}$ a SPSD matrix with largest eigenvalue $\lambda_{\max} = 2$. Consider p_K as above, with $K > \tau/2 - 1$. We have that:*

$$\|h_\tau - p_K\|_\infty \leq 2e^{\frac{(\tau/2)^2}{K+2} - \tau} \frac{(\tau/2)^{K+1}}{K!(K+1-\tau/2)} =: g(K,\tau).$$

Proof. Posing $C = \tau/2$, for $K > C - 1$, we have:

$$\sum_{k=K+1}^{\infty} \frac{C^k}{k!} \leq \frac{1}{K!} \sum_{k=K+1}^{\infty} \frac{C^k}{(K+1)^{k-K}} = \frac{C^K}{K!} \sum_{\ell=1}^{\infty} \frac{C^\ell}{(K+1)^\ell} = \frac{C^{K+1}}{K!(K+1-C)}$$

since $C^2/(K+1) < C$, for $k \geq K+1$, Corollary 2.2 yields:

$$1 \leq d_k \leq \exp(C^2/(K+2)) \leq \exp(C).$$

Since $|T_k(t)| \leq 1$ on $[-1,1]$, again from Corollary 2.2, we obtain:

$$\begin{aligned}\|h_\tau - p_K\|_\infty &= \sup_{\lambda\in[0,\lambda_{\max}]} \left| \sum_{k>K}^{\infty} c_k(\tau)\tilde{T}_k(\lambda) \right| \leq \sum_{k>K}^{\infty} |d_k \bar{c}_k| \\ &\leq \exp\left(\tfrac{C^2}{K+2}\right) 2\exp(-2C) \sum_{k>K}^{\infty} \frac{C^k}{k!} \\ &\leq 2\exp\left(\tfrac{C^2}{K+2} - 2C\right) \frac{C^{K+1}}{K!(K+1-C)}. \qquad \square\end{aligned}$$

While (7) is the approximation error of $\exp(-\tau\mathbf{L})x$, relative to the energy of the signal before diffusion $\|x\|_2^2$, an alternative is to measure this error w.r.t. the energy after diffusion $\| \exp(-\tau\mathbf{L})x\|_2^2$:

$$\eta_K(x) := \frac{\| \exp(-\tau\mathbf{L})x - p_K(\mathbf{L})x\|_2^2}{\| \exp(-\tau\mathbf{L})x\|_2^2}. \tag{8}$$

Since $\| \exp(-\tau\mathbf{L})x\|_2 \geq e^{-\tau\lambda_{\max}}\|x\|_2 = e^{-2\tau}\|x\|_2$, we have $\eta_K(x) \leq \|h_\tau - p_K\|_\infty^2 e^{4\tau}$. Using Lemma 3.1 we obtain for $K > \tau/2 - 1$ and any x:

$$\epsilon_K(x) \leq g^2(K,\tau), \tag{9}$$

$$\eta_K(x) \leq g^2(K,\tau)e^{4\tau}. \tag{10}$$

Specific bounds on relative approximation errors. The bounds in Equations (9) and (10) being worst-case estimates, they may be improved for specific input signals x by taking into account their properties. To illustrate this, let us focus on graph diffusion, with $\mathbf{L}$, a graph Laplacian, and let us moreover assume that $a_1 := \sum_i x_i \neq 0$. Since $a_1/\sqrt{n}$ is the inner product between x and the unit constant vector $(1,\ldots,1)/\sqrt{n}$, which is an eigenvector of the graph Laplacian $\mathbf{L}$ associated to the zero eigenvalue $\lambda_1 = 0$, we have $\| \exp(-\tau\mathbf{L})x\|_2^2 \geq |a_1/\sqrt{n}|^2$. For $K > \tau/2 - 1$ this leads to the specific bound:

$$\eta_K(x) \leq \epsilon_K(x) \frac{\|x\|_2^2}{a_1^2/n} \leq g^2(K,\tau)\frac{n\|x\|_2^2}{a_1^2}, \tag{11}$$

which improves upon (10) if $\tau \geq \frac{1}{4}\log\frac{n\|x\|_2^2}{a_1^2}$.

4. EXPERIMENTS

We apply the proposed polynomial approximation to the diffusion of a graph signal x at scale τ. In general, the largest eigenvalue of the combinatorial graph Laplacian $\mathbf{L}$ is not necessarily $\lambda_{\max} = 2$ (except if we consider the so-called *normalized* graph Laplacian), but we can always return to this generic case, observing that $\exp(-\tau\mathbf{L}) = \exp(-\tau'\mathbf{L}')$ where $\mathbf{L}' = 2\mathbf{L}/\lambda_{\max}$ and $\tau' = \lambda_{\max}\tau/2$. The largest eigenvalue is computed with an Implicitly Restarted Lanczos Method [19]. Then, depending on whether the value of τ' verifies the condition of Eq. (11) or not, the sharpest bound between (10) and (11) will be used to estimate the polynomial degree K that ensures a given approximation accuracy. The recurrence relations (1) is used to compute the action of the polynomials $\tilde{T}_k(\mathbf{L}') = T_k(\mathbf{L}' - \mathbf{Id})$ on x [16], while the coefficients $c_k(\tau')$ derive from the close form expression of (5). All methods are implemented in Python and available in open source for reproducible research [20].

Bound tightness. We compare our new bounds to the tightest bounds we were able to find in the literature [11]:

$$\eta_K(x) \leq 4E(K)^2 \frac{n\|x\|_2^2}{a_1^2}\exp(4\tau) \quad \text{[spec. bound]} \tag{12}$$

$$\leq 4E(K)^2 \quad \text{[gen. bound]} \tag{13}$$

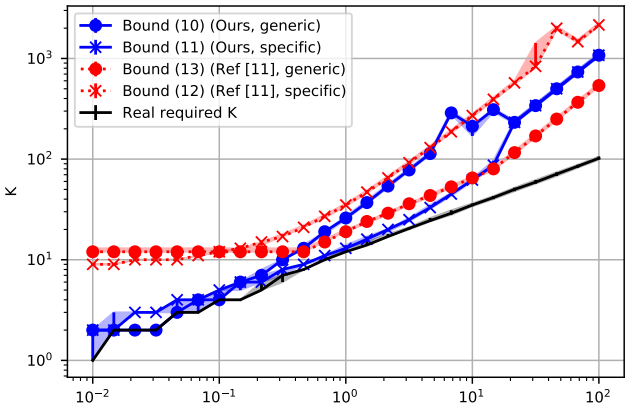

Fig. 1. Minimum degree K to achieve a relative approximation error $\eta_K(x) \leq 10^{-5}$, obtained from the different bounds.

with $E(K) = \begin{cases} e^{\frac{-b(K+1)^2}{2\tau}} \left(1 + \sqrt{\frac{\pi\tau/2}{b}}\right) + \frac{d^{2\tau}}{1-d} & \text{if } K \leq 2\tau \\ \frac{d^K}{1-d} & \text{if } K > 2\tau \end{cases}$

and $b = \frac{2}{1+\sqrt{5}}$ and $d = \frac{\exp(b)}{2+\sqrt{5}}$.

For 25 values of τ from 10^{-2} to 10^2, we considered 100 independant realizations of graph signals, supported on Erdos-Reyni graph of size $n = 200$, with connection probability $p = 5\%$ ($\lambda_{max} \simeq 20$) and whose samples x_i are drawn i.i.d. from a centered standard normal distribution. For each trial, and for each approximation bound, generically noted $B(K, \tau, x)$, we estimate the minimum degree K that ensures $\eta_K(x) \leq B(K, \tau, x) \leq 10^{-5}$, as well as the oracle minimum degree K guaranteeing MSE $\eta_K(x) \leq 10^{-5}$. Corresponding median value and quartiles are plotted against τ in Fig 1.

We observe that our new bounds (blue) follow more closely the true minimum K (black) achieving the targeted precision, up to $\tau \simeq 10$, thus saving computations over the one of [11] (red). Also of interest is the fact that the bounds (11)-(12) specific to the input signal are much tighter than their respective generic counterparts (10)-(13).

Acceleration of multiscale diffusion. Interestingly, when diffusing at multiple scales $\{\tau_1 \cdots \tau_m\}$, computations can be factorized. Indeed, the degree K can be computed only once (using the largest τ_i), as well as the terms $\tilde{T}_k(\mathbf{L}')x$. Note that the matrices $\tilde{T}_k(\mathbf{L}')$ do not need to be stored, but only their action on the signal x, thus avoiding memory issues. Finally, the coefficients c_k's are evaluated for all values τ_i to generate the needed linear combinations of $\tilde{T}_k(\mathbf{L}')x$, $0 \leq k \leq K$. To illustrate this acceleration, our method is compared to `scipy.sparse.linalg.expm_multiply` that uses a Taylor approximation and a linear stepping strategy [13].[2]

In our first experiment, we take the Standford bunny [21], a graph built from a rabbit ceramic scanning (2503 nodes and 65.490 edges, with $\lambda_{max} \simeq 78$). The signal is a Dirac located at a random node. We compute repeatedly the diffusion for 20 scales τ sampled in $[10^{-3}, 10^1]$. Our method is set with a target error $\eta_K \leq 10^{-5}$. When the τ values are linearly spaced, both methods can make use of their respective multiscale acceleration. In this context, our method is about twice faster than `Scipy`: it takes 0.36 s + 6.1×10^{-3} s per scale, while `Scipy` takes 0.74 s + 2.4×10^{-3} s per scale. On the other hand, when the τ values are randomly sampled, `SciPy` cannot make use of its multiscale acceleration and its computation cost increases linearly with the number of τ's, with an average cost of 0.39 s per scale. Whereas, the additional cost for repeating our method for each new τ is negligible (0.0094 s on average) compared to the necessary time to initialize once and for all, the $\tilde{T}_k(\mathbf{L}')x$ (0.30 s).

The trend observed here holds for larger graphs as well. We run a similar experiment on the `ogbn-arxiv` graph from the OGB datasets [22]. We take uniformly sampled scales in $[0.076, 0.24]$ (as in [23]), and set $\eta_K \leq 10^{-3}$. We observe an average computation time of 504 s per scale (1 hr and 24 min for 10 scales) with `Scipy`'s method, and 87 s plus 50 s per scale for our method (around 9 min for 10 scales). If we impose a value $\eta_K \leq 2^{-24}$, comparable to the floating point precision achieved by Scipy, the necessary polynomial degree K only increases by 6%, which does not jeopardise the computational gain of our method. This behavior gives insight into the advantage of using our fast approximation for addressing the multiscale diffusion on very large graphs.

5. CONCLUSION

Our contribution is twofold: first, using the now classical Chebyshev approximation of the exponential function, we significantly improved the state of the art theoretical bound used to determine the minimum polynomial degree needed for an expected approximation error. Second, in the specific case of the heat diffusion kernel applied to a graph structure, we capitalized on the polynomial properties of the Chebyshev coefficients to factorize the calculus of the diffusion operator, reducing thus drastically its computational cost when applied for several values of the diffusion time.

The first contribution is particularly important when dealing with the exponential of extremely large matrices, not necessarily coding for a particular graph. As our new theoretical bound offer the same approximation precision for a polynomial degree downsized by up to one order of magnitude, the computational gain is considerable when modeling the action of operators on large mesh grids, as it can be the case, for instance, in finite element calculus.

Our second input is directly related to our initial motivation in [5] that was to identify the best diffusion time τ in an optimal transport context. Thanks to our accelerated algorithm, we can afford to repeatedly compute the so-called Diffused Wasserstein distance to find the optimal domain adaptation between graphs' measures.

[2] All experiments are in Python using `NumPy` and `SciPy`. They ran on a Intel© Core™ i5-5300U CPU @ 2.30GHz×2 processor and 15.5 GiB RAM on a Linux Mint 20 Cinnamon.

6. REFERENCES

[1] R.M.M. Mattheij, S.W. Rienstra, and J.H.M. Thije Boonkkamp, ten, *Partial differential equations : modeling, analysis, computation*, SIAM monographs on mathematical modeling and computation. SIAM Press, 2005.

[2] Omar De la Cruz Cabrera, Mona Matar, and Lothar Reichel, "Analysis of directed networks via the matrix exponential," *Journal of Computational and Applied Mathematics*, vol. 355, pp. 182–192, 2019.

[3] Hao Zhuang, Shih-Hung Weng, and Chung-Kuan Cheng, "Power grid simulation using matrix exponential method with rational Krylov subspaces," in *2013 IEEE 10th International Conference on ASIC*, 2013.

[4] Maria Pusa and Jaakko Leppänen, "Computing the matrix exponential in burnup calculations," *Nuclear science and engineering*, vol. 164, no. 2, pp. 140–150, 2010.

[5] Amélie Barbe, Marc Sebban, Paulo Gonçalves, Pierre Borgnat, and Rémi Gribonval, "Graph diffusion wasserstein distances," in *ECML PKDD 2020-European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.

[6] Fan R.K. Chung and Fan Chung Graham, *Spectral graph theory*, Number 92. American Math. Soc., 1997.

[7] Mani Mehra, Ankita Shukla, and Günter Leugering, "An adaptive spectral graph wavelet method for PDEs on networks," *Advances in Computational Mathematics*, vol. 47, no. 1, pp. 12, 2021.

[8] Marina Popolizio and Valeria Simoncini, "Acceleration techniques for approximating the matrix exponential operator," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 657–683, 2008.

[9] Mikhail A. Botchev and Leonid A. Knizhnerman, "ART: adaptive residual-time restarting for Krylov subspace matrix exponential evaluations," *Journal of Computational and Applied Mathematics*, vol. 364, 2020.

[10] Vladimir L. Druskin and Leonid A. Knizhnerman, "Two polynomial methods of calculating functions of symmetric matrices," *USSR Computational Mathematics and Mathematical Physics*, vol. 29, no. 6, pp. 112–121, 1989.

[11] Luca Bergamaschi and Marco Vianello, "Efficient computation of the exponential operator for large, sparse, symmetric matrices," *Numerical Linear Algebra with Applications*, vol. 7, no. 1, pp. 27–45, 2000.

[12] John C. Mason and David C. Handscomb, *Chebyshev polynomials*, CRC press, 2002.

[13] Awad H. Al-Mohy and Nicholas J. Higham, "Computing the action of the matrix exponential, with an application to exponential integrators," *SIAM Journal of Scientific Computing*, vol. 33, no. 2, pp. 488–511, 2011.

[14] G.M. Phillips, *Interpolation and Approximation by Polynomials*, CMS Books in Math. Springer, 2003.

[15] Milton Abramowitz, Irene A Stegun, and Robert H Romer, "Handbook of mathematical functions with formulas, graphs, and mathematical tables," 1988.

[16] David I. Shuman, Pierre Vandergheynst, and Pascal Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *2011 IEEE International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, 2011.

[17] Nicholas J. Higham and Awad H. Al-Mohy, "Computing matrix functions," *Acta Numerica*, vol. 19, pp. 159–208, 2010.

[18] David I Shuman, Pierre Vandergheynst, Daniel Kressner, and Pascal Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 736–751, 2018.

[19] Richard B Lehoucq, Danny C Sorensen, and Chao Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, SIAM, 1998.

[20] Sibylle Marcotte, Amélie Barbe, Rémi Gribonval, Titouan Vayer, Marc Sebban, Pierre Borgnat, and Paulo Gonçalves, "Code for reproducible research - Fast Multiscale Diffusion on Graphs," Feb. 2022, code repository available at https://hal.inria.fr/hal-03576498, updates at https://github.com/sibyllema/Fast-Multiscale-Diffusion-on-Graphs.

[21] Robert Riener and Matthias Harders, *Virtual reality in medicine*, Springer Science & Business Media, 2012.

[22] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," 2021, arXiv:2005.00687.

[23] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec, "Learning structural node embeddings via diffusion wavelets," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2018.