A Fast Algorithm for Ranking Users by their Influence in Online Social Platforms

Nouamane Arhachoui, Esteban Bautista, Maximilien Danisch, Anastasios Giovanidis

Sorbonne Université, CNRS – LIP6, Paris, France

November 10th, 2022
ASONAM 2022, Medipol University, Istanbul, Turkey

Outline

Centrality Metrics

SORBONNE UNIVERSITÉ

Centrality metrics are widely used in various applications of Network Science:

- communication networks,
- social networks,
- transportation networks,
- etc.

Some existing metrics

In-degree centrality:

$$C_{in}(v) = \frac{deg^{in}(v)}{|\mathcal{N}| - 1}$$

Betweenness centrality:

$$C_{bet}(v) = \sum_{\substack{s,t \in \mathcal{N} \\ s \neq v \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Closeness centrality:

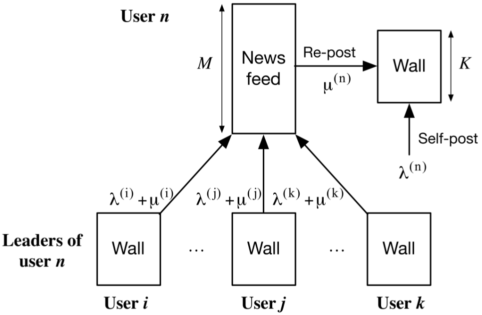
$$C_{clo}(v) = \frac{|\mathcal{N}| - 1}{\sum_{u \in \mathcal{N}} d(v, u)}$$

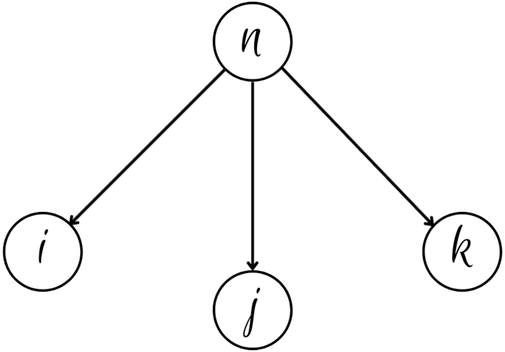
PageRank:

$$\pi(v, \alpha) = \alpha \sum_{\substack{u \in \mathcal{N} \\ u \neq v}} A_{uv} \frac{\pi(u, \alpha)}{\max(deg^{out}(u), 1)} + \frac{1 - \alpha}{|\mathcal{N}|}$$

Existing centrality metrics do not consider user activity within the network.

Social Platform Model





- $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $(i, j) \in \mathcal{E}$ iff user i follows user j. $|\mathcal{N}| = N$.
- Each user has a set of **followers** $\mathcal{F}^{(n)}$ and a set of **leaders** $\mathcal{L}^{(n)}$.
- Each user has 2 queues: a **Wall** of size K and a **Newsfeed** of size M.
- Each user has a **posting rate** $\lambda^{(n)}$ (posts created by n per unit of time) and a **re-posting rate** $\mu^{(n)}$ (posts that n shares per unit of time).

Focus on posts of origin i

SORBONNE UNIVERSITÉ

Presence on Newsfeeds

$\mathbf{p}_i = (p_i^{(1)} \; p_i^{(2)} \; \cdots \; p_i^{(N)})^T$
$\forall n \in \mathcal{N}$, $p_i^{(n)}$ is the expected percentage of posts originating from user i on the news-feed of user n

Presence on Walls

$\mathbf{q}_i = (q_i^{(1)} \; q_i^{(2)} \; \cdots \; q_i^{(N)})^T$
$\forall n \in \mathcal{N}$, $q_i^{(n)}$ is the expected percentage of posts originating from user i on the wall of user n (**influence of i on n**)

The ψ-score

The influence of a user i over the entire network is:

$$\psi_i = \frac{1}{N}\sum_{n=1}^{N} q_i^{(n)} = \frac{1}{N}\mathbf{1}^T\mathbf{q}_i$$

Focus on posts of origin i

SORBONNE UNIVERSITÉ

Presence on Newsfeeds

$\mathbf{p}_i = (p_i^{(1)} \; p_i^{(2)} \; \cdots \; p_i^{(N)})^T$

$\forall n \in \mathcal{N}$, $p_i^{(n)}$ is the expected percentage of posts originating from user i on the news-feed of user n

Presence on Walls

$\mathbf{q}_i = (q_i^{(1)} \; q_i^{(2)} \; \cdots \; q_i^{(N)})^T$

$\forall n \in \mathcal{N}$, $q_i^{(n)}$ is the expected percentage of posts originating from user i on the wall of user n (**influence of i on n**)

The ψ-score

The influence of a user i over the entire network is:

$$\psi_i = \frac{1}{N} \sum_{n=1}^{N} q_i^{(n)} = \frac{1}{N} \mathbf{1}^T \mathbf{q}_i$$

Relation with PageRank

SORBONNE UNIVERSITÉ

Theorem

When all the users have the same activity, i.e. $\forall n \in [\![1, N]\!]$ $\lambda^{(n)} = \lambda$ and $\mu^{(n)} = \mu$ and if $\frac{\mu}{\lambda+\mu} = \alpha \in [0, 1]$, then ψ-score $=$ PageRank with a damping factor α

- ψ-score uses additional information useful for measuring the influence
- In social networks, users have heterogeneous behaviors (i.e. different λ and μ)

Compute the ψ-score

SORBONNE UNIVERSITÉ

Process to compute ψ_i

Solve the following linear systems:

$$\mathbf{p}_i = \mathbf{A}\mathbf{p}_i + \mathbf{b}_i$$
$$\mathbf{q}_i = \mathbf{C}\mathbf{p}_i + \mathbf{d}_i$$

and use $\psi_i = \frac{1}{N}\sum_{n=1}^{N} q_i^{(n)}$

where,

$\mathbf{A}$	$a_{ji} = \frac{\mu^{(i)}}{\sum_{\ell \in \mathscr{L}^{(j)}} (\lambda^{(\ell)}+\mu^{(\ell)})} \mathbb{1}_{\{i \in \mathscr{L}^{(j)}\}}$	$\mathbf{b}_i$	$b_{ji} = \frac{\lambda^{(i)}}{\sum_{\ell \in \mathscr{L}^{(j)}} (\lambda^{(\ell)}+\mu^{(\ell)})} \mathbb{1}_{\{i \in \mathscr{L}^{(j)}\}}$
$\mathbf{C}$	$c_{ji} = \frac{\mu^{(j)}}{\lambda^{(j)}+\mu^{(j)}} \mathbb{1}_{\{j=i\}}$	$\mathbf{d}_i$	$d_{ji} = \frac{\lambda^{(i)}}{\lambda^{(i)}+\mu^{(i)}} \mathbb{1}_{\{j=i\}}$

To have the whole ψ-score vector, the process needs to be executed for each user in the network. This represents solving N systems of equations.

Compute the ψ-score

SORBONNE UNIVERSITÉ

Process to compute ψ_i

Solve the following linear systems:

$$\mathbf{p}_i = \mathbf{A}\mathbf{p}_i + \mathbf{b}_i$$
$$\mathbf{q}_i = \mathbf{C}\mathbf{p}_i + \mathbf{d}_i$$

and use $\psi_i = \frac{1}{N}\sum_{n=1}^{N} q_i^{(n)}$

where,

$\mathbf{A}$	$a_{ji} = \frac{\mu^{(i)}}{\sum_{\ell \in \mathscr{L}^{(j)}} (\lambda^{(\ell)} + \mu^{(\ell)})} \mathbb{1}_{\{i \in \mathscr{L}^{(j)}\}}$	$\mathbf{b}_i$	$b_{ji} = \frac{\lambda^{(i)}}{\sum_{\ell \in \mathscr{L}^{(j)}} (\lambda^{(\ell)} + \mu^{(\ell)})} \mathbb{1}_{\{i \in \mathscr{L}^{(j)}\}}$
$\mathbf{C}$	$c_{ji} = \frac{\mu^{(j)}}{\lambda^{(j)} + \mu^{(j)}} \mathbb{1}_{\{j = i\}}$	$\mathbf{d}_i$	$d_{ji} = \frac{\lambda^{(i)}}{\lambda^{(i)} + \mu^{(i)}} \mathbb{1}_{\{j = i\}}$

To have the whole ψ-score vector, the process needs to be executed for each user in the network. This represents solving N systems of equations.

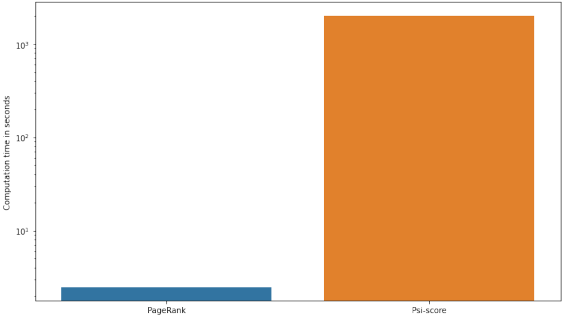
Outline

SORBONNE UNIVERSITÉ

Problem statement

Problem

- The current computation of the ψ-score vector is too slow (compared e.g. to PageRank)
- There are a linear system for each user in the network
- Solving N systems of N equations is required to get the ψ-score vector



Problem Statement

Given a social graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where the nodes have a posting and sharing activity, we aim for an algorithm that computes the ψ-score for all nodes as fast as PageRank.

Outline

First step: Rewrite the system

Instead of solving these N systems of N equations:

$$\mathbf{p}_i = \mathbf{A}\mathbf{p}_i + \mathbf{b}_i$$
$$\mathbf{q}_i = \mathbf{C}\mathbf{p}_i + \mathbf{d}_i$$

we can rewrite everything as follows:

$$\mathbf{P} = \mathbf{A}\mathbf{P} + \mathbf{B}$$
$$\mathbf{Q} = \mathbf{C}\mathbf{P} + \mathbf{D}$$

with,

$$\mathbf{P} = \begin{pmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_N\end{pmatrix}$$
$$\mathbf{Q} = \begin{pmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_N\end{pmatrix}$$
$$\mathbf{B} = \begin{pmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_N\end{pmatrix}$$
$$\mathbf{D} = \begin{pmatrix}\mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_N\end{pmatrix}$$

all square matrices.

Theorem

$\mathbf{A}$ is sub-stochastic[1] $\Rightarrow \rho(\mathbf{A}) < 1$

$$\Rightarrow \mathbf{P} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \sum_{t=0}^{\infty} \mathbf{A}^t\mathbf{B}$$

[1] A sub-stochastic matrix is a real square matrix having each row summing to a value strictly lower than 1.

Second step: Get directly the ψ-score vector

In the same fashion, computing:

$$\forall i, \quad \psi_i = \frac{1}{N}\mathbf{1}^T\mathbf{q}_i$$

becomes:

$$\boldsymbol{\psi}^T = \frac{1}{N}\mathbf{1}^T\mathbf{Q}$$

where,

$$\boldsymbol{\psi} = \begin{pmatrix}\psi_1 & \psi_2 & \cdots & \psi_N\end{pmatrix}^T$$

is the ψ-score vector

Power-ψ

With all these changes, we obtain:

$$\boldsymbol{\psi}^T = \frac{1}{N}\left(\mathbf{s}^T\mathbf{B} + \mathbf{1}^T\mathbf{D}\right)$$

where,

$$\mathbf{s}^T = \sum_{t=0}^{\infty} \mathbf{1}^T\mathbf{C}\mathbf{A}^t$$

$\mathbf{C}$ and $\mathbf{D}$ are diagonal matrices. No need to compute their product with $\mathbf{1}$. Let $\mathbf{c} := (\mathbf{1}^T\mathbf{C})^T$ and $\mathbf{d} := (\mathbf{1}^T\mathbf{D})^T$

Approximating the sum $\mathbf{s}$

$$\mathbf{s}_t^T = \sum_{k=0}^{t} \mathbf{c}^T \mathbf{A}^k, \qquad \mathbf{s} = \lim_{t\to\infty} \mathbf{s}_t,$$

The sub-stochasticity of $\mathbf{A}$ ensures the convergence of $\mathbf{s}$.

Approximating the sum $\mathbf{s}$

SORBONNE UNIVERSITÉ

Truncating the sum gives us the following recursive expression of it:

$$\mathbf{s}_t^T = \mathbf{s}_{t-1}^T \mathbf{A} + \mathbf{c}^T$$

where $\mathbf{s}_0 = \mathbf{c}$.

gap parameter computed at each iteration to check the convergence:

$$\varepsilon_t = \left\| \mathbf{s}_t^T - \mathbf{s}_{t-1}^T \right\|$$

Algorithm

SORBONNE UNIVERSITÉ

Algorithm 1: `Power-`ψ**:** Power iteration based algorithm for the ψ-score vector.

input : N number of users, $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$, two vectors $\mathbf{c}$ and $\mathbf{d}$, $\mathbf{s}$-tolerance ε

output: vector $\boldsymbol{\psi}$ with the ψ-score of all users

$\mathbf{s} \leftarrow \mathbf{c}$;

$B\_norm \leftarrow \|\mathbf{B}\|$;

$t \leftarrow 0$;

$gap \leftarrow 1$;

while $(gap > \varepsilon)$ **do**

- $\mathbf{s}_{old} \leftarrow \mathbf{s}$;
- $\mathbf{s}^T \leftarrow \mathbf{s}_{old}^T \mathbf{A} + \mathbf{c}$;
- $gap \leftarrow B\_norm \, \|\mathbf{s}_{old} - \mathbf{s}\|$;
- $t \leftarrow t + 1$;

end

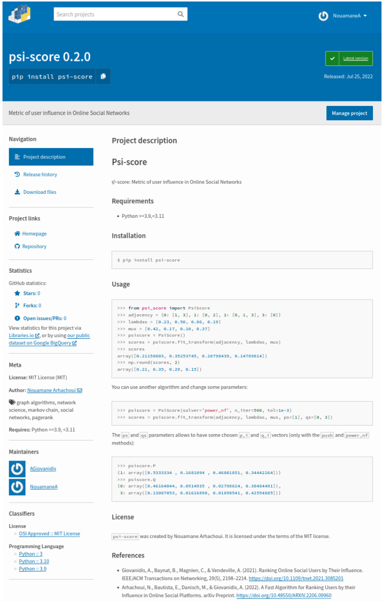
$\boldsymbol{\psi}^T \leftarrow \frac{1}{N}\left(\mathbf{s}^T \mathbf{B} + \mathbf{d}^T\right)$;

return $\boldsymbol{\psi}$;

Outline

SORBONNE UNIVERSITÉ

The *psi-score* Python package



Psi-score

ψ-score: Metric of user influence in Online Social Networks

Requirements

- Python >=3.9,<3.11

Installation

```
$ pip install psi-score
```

Usage

```
>>> from psi_score import PsiScore
>>> adjacency = {0: [1, 3], 1: [0, 2], 2: [0, 1, 3], 3: [0]}
>>> lambdas = [0.23, 0.50, 0.86, 0.19]
>>> mus = [0.42, 0.17, 0.10, 0.37]
>>> psiscore = PsiScore()
>>> scores = psiscore.fit_transform(adjacency, lambdas, mus)
>>> scores
array([0.21158803, 0.35253745, 0.28798439, 0.14789014])
>>> np.round(scores, 2)
array([0.21, 0.35, 0.29, 0.15])
```

You can use another algorithm and change some parameters:

```
>>> psiscore = PsiScore(solver='power_nf', n_iter=500, tol=1e-3)
>>> scores = psiscore.fit_transform(adjacency, lambdas, mus, ps=[1], qs=[0, 3])
```

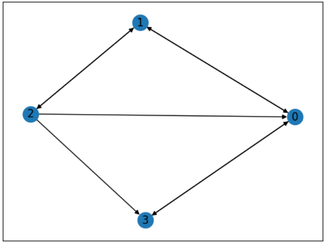
ψ-score

```
[1]: from psi_score import PsiScore
     import networkx as nx
```

```
[2]: adjacency = {0: [1, 3], 1: [0, 2], 2: [0, 1, 3], 3: [0]}
     lambdas = [0.23, 0.50, 0.86, 0.19]
     mus = [0.42, 0.17, 0.10, 0.37]
```

```
[3]: G = nx.DiGraph(adjacency)
```

```
[4]: nx.draw_networkx(G)
```




```
[5]: power_psi = PsiScore()
     scores = power_psi.fit_transform(adjacency, lambdas, mus)

     100% (500 of 500) |######################| Elapsed Time: 0:00:00 Time:  0:00:00
```

```
[6]: import numpy as np
     np.round(scores, 2)

[6]: array([0.21, 0.35, 0.29, 0.15])
```

```
[7]: power_nf = PsiScore(solver='power_nf', n_iter=500, tol=1e-3)
     scores = power_nf.fit_transform(adjacency, lambdas, mus, ps=[1], qs=[0, 3])

     100% (4 of 4) |##########################| Elapsed Time: 0:00:00 Time:  0:00:00
```

```
[8]: power_nf.P

[8]: {1: array([0.5333334 , 0.1681094 , 0.46801851, 0.34442264])}
```

PageRank

```
[10]: pr = list(nx.pagerank(G).values())
      pr = np.array(pr)
```

```
[11]: # Homogeneous activity
      lambdas = [0.15]*len(G)
      mus = [0.85]*len(G)
```

```
[12]: psi_homogeneous = PsiScore(solver='power_psi')
      psi_homogeneous.fit(adjacency, lambdas, mus)

      100% (500 of 500) |######################| Elapsed Time: 0:00:00 Time:  0:00:00

[12]: <psi_score.psi_score.PsiScore at 0x7f425db554c0>
```

```
[13]: print('PageRank vector:')
      print(np.round(pr, 3))

      print()
      print('Psi-score vector (for homogeneous activity):')
      print(np.round(psi_homogeneous.scores, 3))

      print()
      print('Psi-score vector (for heterogeneous activity):')
      print(np.round(power_psi.scores, 3))

      PageRank vector:
      [0.382 0.239 0.139 0.239]

      Psi-score vector (for homogeneous activity):
      [0.382 0.239 0.139 0.239]

      Psi-score vector (for heterogeneous activity):
      [0.212 0.353 0.288 0.148]
```

Outline

Numerical Analysis

Datasets:

Dataset name	Type	#Nodes	#Edges
DBLP	Citation Network	12 591	49 743
Twitter	Social Network	465 017	834 797
Facebook	Social Network	63 731	817 035
HepPh arXiv	Citation Network	34 546	421 578

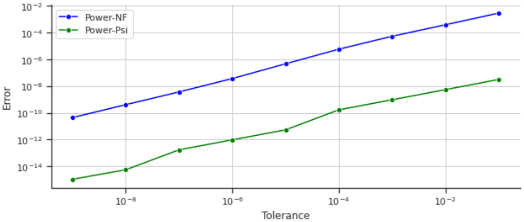
Three types of experiments:

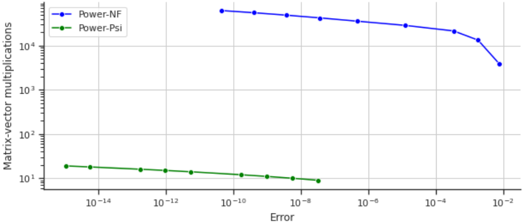
- Precision assessment for a given tolerance criterion
- Performance assessment for a measured error
- Speed assessment for a given tolerance (unable to measure the error for large datasets)

Two scenarios:

- heterogeneous activity scenario: users do not necessarily have the same posting and re-posting activity
- homogeneous activity scenario: all users have the same activity (i.e. the same λ and μ); in this case the ψ-score is exactly PageRank with $\alpha = \frac{\mu}{\lambda+\mu}$

(i) Heterogeneous scenario

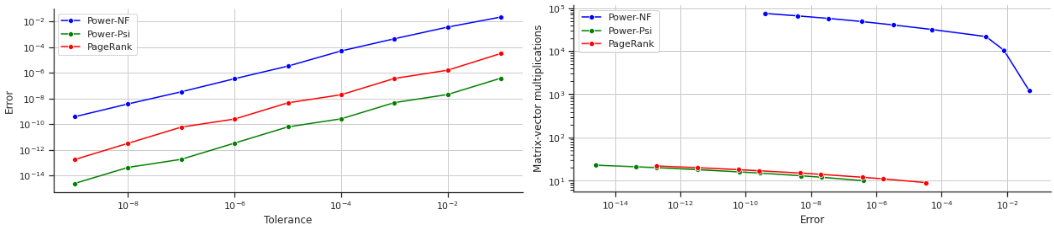




Dataset	Power-NF	Power-ψ
DBLP	17.805 sec	0.029 sec
Facebook	1764.226 sec	0.307 sec
Twitter	14526.039 sec	0.634 sec
HepPh	272.358 sec	0.622 sec

with $\varepsilon = 10^{-9}$

(ii) Homogeneous scenario



Dataset	PageRank	Power-NF	Power-ψ
DBLP	0.023 sec	20.775 sec	0.034 sec
Facebook	0.308 sec	2253.302 sec	0.454 sec
Twitter	0.584 sec	17411.146 sec	0.806 sec
HepPh	0.361 sec	360.769 sec	0.908 sec

with $\varepsilon = 10^{-9}$

Conclusion and Future Work

The proposed method

- is nearly as fast as PageRank
- outperforms the state-of-the-art alternative
- enables scalability for real-world datasets

Future work:

- Explore generalizations of the ψ-score in time evolving networks
- Study the effect of day/night (or ON/OFF) user activity on the model

References I

- Arhachoui, Nouamane et al. (2022). *A Fast Algorithm for Ranking Users by their Influence in Online Social Platforms*. doi: 10.48550/ARXIV.2206.09960. url: https://arxiv.org/abs/2206.09960.
- Giovanidis, Anastasios et al. (2021). "Ranking Online Social Users by Their Influence". In: *IEEE/ACM Trans. Netw.* 29.5, pp. 2198–2214. doi: 10.1109/TNET.2021.3085201. url: https://doi.org/10.1109/TNET.2021.3085201.
- Wan, Zelin et al. (2021). "A Survey on Centrality Metrics and Their Network Resilience Analysis". In: *IEEE Access* 9, pp. 104773–104819. doi: 10.1109/ACCESS.2021.3094196.

Appendix A: Obtaining the Power-ψ equation

$$
\begin{aligned}
\boldsymbol{\psi}^T &= \frac{1}{N}\mathbf{1}^T\mathbf{Q} \\
&= \frac{1}{N}\mathbf{1}^T(\mathbf{CP}+\mathbf{D}) \\
&= \frac{1}{N}\mathbf{1}^T\left[\mathbf{C}(\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}+\mathbf{D}\right] \\
&= \frac{1}{N}\mathbf{1}^T\left[\mathbf{C}\left(\sum_{t=0}^{\infty}\mathbf{A}^t\right)\mathbf{B}+\mathbf{D}\right] \\
&= \frac{1}{N}\left[\mathbf{1}^T\mathbf{C}\left(\sum_{t=0}^{\infty}\mathbf{A}^t\right)\mathbf{B}+\mathbf{1}^T\mathbf{D}\right] \\
&= \frac{1}{N}\left[\left(\sum_{t=0}^{\infty}\mathbf{1}^T\mathbf{C}\mathbf{A}^t\right)\mathbf{B}+\mathbf{1}^T\mathbf{D}\right]
\end{aligned}
$$

The way s is truncated impacts the precision of ψ.

To overcome this, let us define another gap:

$$\delta_t = \left\| \psi_t^T - \psi_{t-1}^T \right\|$$

We can now compare the two:

$$\begin{aligned}
\psi_t^T - \psi_{t-1}^T &= \frac{1}{N}(\mathbf{s}_t^T \mathbf{B} + \mathbf{d}^T) - \frac{1}{N}(\mathbf{s}_{t-1}^T \mathbf{B} + \mathbf{d}^T) \\
&= \frac{1}{N}(\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B}
\end{aligned}$$

$$\begin{aligned}
\delta_t &= \frac{1}{N} \left\| (\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B} \right\| \\
&\leq \frac{1}{N} \left\| \mathbf{s}_t^T - \mathbf{s}_{t-1}^T \right\| \left\| \mathbf{B} \right\| \\
\delta_t &\leq \frac{\varepsilon_t \left\| \mathbf{B} \right\|}{N}
\end{aligned}$$

Setting the termination condition to $\varepsilon_t \left\| \mathbf{B} \right\| \leq \varepsilon$ ensures that $\delta_t \leq \frac{\varepsilon}{N} \leq \varepsilon$

The way s is truncated impacts the precision of ψ.

To overcome this, let us define another gap:

$$\delta_t = \left\| \boldsymbol{\psi}_t^T - \boldsymbol{\psi}_{t-1}^T \right\|$$

We can now compare the two:

$$\begin{aligned}
\boldsymbol{\psi}_t^T - \boldsymbol{\psi}_{t-1}^T &= \frac{1}{N}(\mathbf{s}_t^T \mathbf{B} + \mathbf{d}^T) - \frac{1}{N}(\mathbf{s}_{t-1}^T \mathbf{B} + \mathbf{d}^T) \\
&= \frac{1}{N}(\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B} \\
\delta_t &= \frac{1}{N} \left\| (\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B} \right\| \\
&\leq \frac{1}{N} \left\| \mathbf{s}_t^T - \mathbf{s}_{t-1}^T \right\| \left\| \mathbf{B} \right\| \\
\delta_t &\leq \frac{\varepsilon_t \left\| \mathbf{B} \right\|}{N}
\end{aligned}$$

Setting the termination condition to $\varepsilon_t \left\| \mathbf{B} \right\| \leq \varepsilon$ ensures that $\delta_t \leq \frac{\varepsilon}{N} \leq \varepsilon$

The way s is truncated impacts the precision of ψ.

To overcome this, let us define another gap:

$$\delta_t = \left\| \psi_t^T - \psi_{t-1}^T \right\|$$

We can now compare the two:

$$\begin{aligned}
\psi_t^T - \psi_{t-1}^T &= \frac{1}{N}(\mathbf{s}_t^T \mathbf{B} + \mathbf{d}^T) - \frac{1}{N}(\mathbf{s}_{t-1}^T \mathbf{B} + \mathbf{d}^T) \\
&= \frac{1}{N}(\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B} \\
\delta_t &= \frac{1}{N} \left\| (\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B} \right\| \\
&\leq \frac{1}{N} \left\| \mathbf{s}_t^T - \mathbf{s}_{t-1}^T \right\| \left\| \mathbf{B} \right\| \\
\delta_t &\leq \frac{\varepsilon_t \left\| \mathbf{B} \right\|}{N}
\end{aligned}$$

Setting the termination condition to $\varepsilon_t \left\| \mathbf{B} \right\| \leq \varepsilon$ ensures that $\delta_t \leq \frac{\varepsilon}{N} \leq \varepsilon$