GraphCPP: A Data-Driven System for Concurrent Point-to-Point Queries in Dynamic Graphs

IEEE Publication Technology Department

***Abstract*—With the widespread application of graph processing techniques in areas such as map navigation and network analysis, there is a growing demand for high throughput in handling numerous point-to-point query tasks concurrently on the same underlying graph. However, existing graph query systems have primarily focused on optimizing the speed of individual point-to-point queries. When it comes to concurrent graph computations, these systems suffer from poor overall throughput due to redundant data access overhead and computational costs.**

We observe that due to the power-law distribution characteristic of graph data, the traversal paths of different queries often overlap on local paths composed of a small number of high-degree vertices, demonstrating the data similarity of concurrent point-to-point query tasks. This inspires us to propose a data-driven concurrent point-to-point query system - GraphCPP. It employs a partitioning approach to organize the graph structure data into LLC-level blocks. A key feature is the prioritized loading of active blocks, determined by the number of associated active vertices. This loading strategy, by focusing on active blocks, is instrumental in facilitating concurrent execution of associated query tasks. The outcome is enhanced data sharing and improved data access efficiency. Additionally, GraphCPP incorporates a core subgraph mechanism, which plays a critical role in promoting computational sharing. This mechanism is dedicated to pre-calculating distance values between high-degree vertices within the graph. This pre-calculation enables the rapid determination of distance values for frequently shared path segments when queries are initiated. In this way, it expedites the convergence of query results while optimizing computational sharing. Furthermore, GraphCPP employs predictive techniques during task scheduling to harness data similarity within concurrent point-to-point query tasks effectively. By batching and selecting similar tasks from the task pool, it maximizes the utilization of data similarity. We compare GraphCPP with state-of-the-art point-to-point query systems, including SGraph[x], Tripoline[x], and Pnp[x]. Experimental results demonstrate that GraphCPP improves the efficiency of concurrent point-to-point queries by a factor of xxxx

***Index Terms*—Class, IEEEtran, LATEX, paper, style, template, typesetting.**

I. Introduction

POINT-TO-POINT query tasks on graphs refer to the exploration of a specific relationship between two objects utilizing the graph as a universal data structure. Unlike traditional graph query methods, point-to-point queries on graphs specifically analyze the associations or paths between two specific vertices, without the need to consider complex queries involving the entire graph or its large-scale subsets. This targeted querying strategy endows point-to-point queries with significant optimization potential. For certain versions of monotonic graph query algorithms, such as Point-to-Point Shortest Path for SSSP (PPSP), Point-to-Point Widest Path for SSWP (PPWP), and Point-to-Point Narrowest Path for SSNP (PPNP), specific path attributes between two vertices can be accurately determined without the need for or with minimal querying and processing of unrelated other vertices or edges. Due to the efficiency of point-to-point queries in graph analysis, it has found extensive practical applications in various fields. For instance, in logistics and transportation, finding the shortest path between two locations; in social network analysis, recommending potential friends to users by examining the relationship chain between two users; in financial risk analysis, analyzing how risks propagate from one entity to another; these popular applications have raised the demand for executing large-scale concurrent point-to-point queries on the same underlying graph.

However, existing solutions for point-to-point queries have focused on accelerating the efficiency of individual queries, overlooking optimization for concurrent queries. To achieve concurrent point-to-point queries, the following two challenges need to be addressed.

Firstly, achieving data sharing is imperative. There exists significant overlap in the traversal paths of different query tasks. However, under the existing execution paradigm, data isolation between concurrent tasks prevents the sharing of overlapping data, resulting in redundant data access. Additionally, different tasks exhibit varying access sequences for the same graph structure data, further complicating the facilitation of data sharing.

Secondly, enabling computational sharing is crucial. Graph data often adheres to a power-law distribution, where segments formed by a small number of high-degree vertices frequently appear in the shortest paths of different queries. Due to the abundance of neighboring vertices surrounding high-degree vertices, repeated traversals by different tasks often lead to an explosive growth in computational costs. Some existing systems have attempted to employ global indexes for computational sharing, incurring substantial costs in computation, storage, and updates. This approach limits the coverage and precision of computational sharing.

In response to the previously mentioned challenges, we introduce GraphCPP, a data-driven system tailored for concurrent point-to-point queries on dynamic graphs. To address the issue of data sharing among concurrent tasks, we present a data-driven caching execution mechanism that shifts from the conventional "task→data" scheduling approach to a "data→task" strategy. This change allows for concurrent access to graph structure data across multiple tasks. Under this execution paradigm, GraphCPP initially determines the order of data scheduling, dividing graph structure data into fine-grained blocks at the LLC level. Subsequently, it associates each query task with the relevant graph block based on the block where the active vertex set of the task resides. As the active vertices change in each round, the number of associated tasks for shared blocks is updated accordingly, with blocks having more associated tasks being given higher scheduling priority. To implement the "data→task" scheduling approach, GraphCPP utilizes an associated task triggering mechanism. It prioritizes the loading of graph blocks into the LLC and utilizes task-data association information obtained in each round to trigger batch task execution associated with the current block, enhancing efficient access to shared data. In response to the challenge of computational sharing, GraphCPP introduces a query acceleration mechanism based on core subgraphs. It streamlines the traditional "global index," which maintains distance values for all vertices, into a "core subgraph index" that exclusively maintains distance values between high-degree vertices. The core subgraph effectively creates direct edges between interconnected high-degree vertices, representing the shortest distance between them. When querying a high-degree vertex, the program can access all other high-degree vertices, similar to visiting neighboring nodes, enabling computational sharing across overlapping paths. The streamlined core subgraph index incurs significantly lower overhead than the global index, allowing for the inclusion of more high-degree vertices in the core subgraph. This expands the coverage of frequently shared paths, ultimately enhancing computational sharing performance. Additionally, by predicting the traversal paths of different query tasks, we prioritize batch task execution for tasks with substantial overlap, further optimizing the performance of concurrent queries.

This paper makes the following contributions:

1) Analyzed the performance bottleneck caused by redundant data access in existing point-to-point query systems when handling concurrent point-to-point query tasks. Proposed leveraging data access similarity among concurrent query tasks to optimize concurrent task throughput.
2) Developed GraphCPP, a data-driven concurrent point-to-point query system on dynamic graphs, achieving data and computational sharing among concurrent tasks. Additionally, introduced a strategy for batch execution of similar tasks.
3) We compared GraphCPP with the state-of-the-art point-to-point query system XXXXXX. The results demonstrate XXXXXXXXXX.

II. BACKGROUND AND MOTIVATION

Existing solutions have primarily focused on accelerating the speed of individual queries. For instance, PnP employs a lower-bound-based pruning method to reduce redundant access during the query process. Tripoline maintains a daily index from the central vertex to other vertices, enabling rapid queries without prior knowledge. SGraph leverages the principle of triangular inequalities and proposes a "upper bound + lower bound" pruning method, further reducing redundant access during point-to-point query processes. However, as shown in Figure x, our statistics indicate that concurrent point-to-point queries on graphs are becoming an increasingly pressing demand. They prioritize the throughput of concurrent query tasks and are more tolerant of the speed of individual queries. As depicted in Figure x, we demonstrate that existing systems exhibit poor throughput when handling large-scale concurrent queries. This undesirable outcome arises from the substantial redundant data access between concurrent tasks. To qualitatively analyze the aforementioned issues, we conducted performance evaluations of parallel point-to-point queries on XXXXX (machine configurations) using XXXXX (existing best practices) on XXXXX (graph dataset).

This chapter is divided into three parts. We first introduce some concepts in concurrent point-to-point queries. Next, we analyze the performance bottlenecks of current point-to-point query schemes when handling concurrent tasks. Finally, we present the insights obtained from our observations and analysis.

A. Preliminaries

Definition 1: Graph. We represent a directed graph as $G = (V, E)$, where V is the set of vertices and E is the set of directed edges composed of vertices in V (edges in an undirected graph can be split into directed edges in two different directions). We use $|V|$ and $|E|$ to respectively denote the number of vertices and edges.

Definition 2: Graph Partition. We use $P_i = (V_{P_i}, E_{P_i})$ to denote the i-th graph partition of a directed graph, where V_{P_i} represents the set of vertices in the graph partition, and E_{P_i} is the set of directed edges composed of vertices in V_{P_i}. In a distributed system, different machine-specific graph partitions P_i are distinct. We partition the graph using edge cuts, where the same vertex may appear on different computing nodes, but there is only one primary vertex, while the others are mirror vertices.

Definition 3: Point-to-Point Query. We use $q_i = (s_i, d_i)$ to represent the query corresponding to task i. Here, s_i and d_i respectively denote the source and destination vertices of query q_i. The result value obtained by query q_i is represented as $R_{s,d}$. For different algorithms, it holds different meanings. For example, for shortest path queries, R_{ib} represents the shortest path between s_i and d_i. We use $Q = q_1, q_2, \ldots, q_{|Q|}$ to represent the set of concurrent point-to-point queries, where $|Q|$ denotes the total number of queries.

Definition 4: Index: An index records the distance from one vertex to other vertices and achieves computational sharing by calculating frequently accessed paths. Global Index: We select the top k vertices with the highest degrees in the graph as index vertices h_i (where $i \in [1, k]$, and users can specify k according to their needs, typically set to 16). In this context, $d_{i,j}$ (where $v_j \in V$) represents the distance from index vertex

h_i to any vertex v_j in the graph. If there is no reachable path between two vertices, the value is set to an extremely high value. Similarly, $d_{j,i}$ (where $v_j \in V$) represents the distance from any vertex v_j in the graph to the index vertex h_i. In undirected graphs, $d_{i,j}$ and $d_{j,i}$ are equal. The creation of this global index is designed to meet specific requirements. Core Subgraph Index: We choose highly connected vertices h_j in the degree range (k, m) to establish a core subgraph index (where $j \in (k, m]$, and users can specify m based on their requirements, typically one order of magnitude larger than k).

Definition 5: Upper Bound and Lower Bound: In point-to-point queries, the upper bound (UB) represents the known shortest distance value from the source vertex to the destination vertex. The lower bound (LB) for the current vertex v to the destination vertex is a conservative estimate of the shortest distance. The predicted LB is less than or equal to the actual shortest distance from vertex v to the destination vertex. According to the triangle inequality on the graph, if a path's distance is greater than UB or if adding the value of LB makes it greater than UB, then this path is certainly worse than existing paths and should be pruned. The values of upper and lower bounds need to be derived with the help of an index. Essentially, they are a form of computation sharing.

Definition 6: Core Subgraph. Similar to an index, a core subgraph also identifies highly connected vertices in a graph, but it employs a lower threshold for selection, which means that more vertices can be chosen. These highly connected vertices form the core subgraph, where the edge weights between two high-degree vertices represent the distance values between the two points. If two vertices are ultimately unreachable, the edge weight is set to a very large value. The key distinction between the core subgraph and a global index is that the core subgraph only maintains indices among high-degree vertices and does not store distance values for reaching non-high-degree vertices.

III. LATEX Distributions: Where to Get Them

IEEE recommends using the distribution from the TEXUser Group at http://www.tug.org. You can join TUG and obtain a DVD distribution or download for free from the links provided on their website: http://www.tug.org/texlive/. The DVD includes distributions for Windows, Mac OS X and Linux operating systems.

IV. Where to get the IEEEtran Templates

The **IEEE Template Selector** will always have the most up-to-date versions of the LATEX and MSWord templates. Please see: https://template-selector.ieee.org/ and follow the steps to find the correct template for your intended publication. Many publications use the IEEETran LaTeX templates, however, some publications have their own special templates. Many of these are based on IEEEtran, but may have special instructions that vary slightly from those in this document.

V. Where to get LATEX help - user groups

The following on-line groups are very helpful to beginning and experienced LATEX users. A search through their archives can provide many answers to common questions.

http://www.latex-community.org/
https://tex.stackexchange.com/

VI. Document Class Options in IEEEtran

At the beginning of your LATEX file you will need to establish what type of publication style you intend to use. The following list shows appropriate documentclass options for each of the types covered by IEEEtran.

Regular Journal Article
`\documentclass[journal]IEEEtran`

Conference Paper
`\documentclass[conference]IEEEtran`

Computer Society Journal Article
`\documentclass[10pt,journal,compsoc]IEEEtran`

Computer Society Conference Paper
`\documentclass[conference,compsoc]IEEEtran`

Communications Society Journal Article
`\documentclass[journal,comsoc]IEEEtran`

Brief, Correspondence or Technote
`\documentclass[9pt,technote]IEEEtran`

There are other options available for each of these when submitting for peer review or other special requirements. IEEE recommends to compose your article in the base 2-column format to make sure all your equations, tables and graphics will fit the final 2-column format. Please refer to the document "IEEEtran_HOWTO.pdf" for more information on settings for peer review submission if required by your EIC.

VII. How to Create Common Front Matter

The following sections describe general coding for these common elements. Computer Society publications and Conferences may have their own special variations and will be noted below.

A. Paper Title

The title of your paper is coded as:

```
\title{The Title of Your Paper}
```

Please try to avoid the use of math or chemical formulas in your title if possible.

B. Author Names and Affiliations

The author section should be coded as follows:

```
\author{Masahito Hayashi
\IEEEmembership{Fellow, IEEE}, Masaki Owari
\thanks{M. Hayashi is with Graduate School
of Mathematics, Nagoya University, Nagoya,
Japan}
\thanks{M. Owari is with the Faculty of
Informatics, Shizuoka University,
```

```
Hamamatsu, Shizuoka, Japan.}
}
```

Be sure to use the \IEEEmembership command to identify IEEE membership status. Please see the "IEEEtran_HOWTO.pdf" for specific information on coding authors for Conferences and Computer Society publications. Note that the closing curly brace for the author group comes at the end of the thanks group. This will prevent you from creating a blank first page.

C. Running Heads

The running heads are declared by using the `\markboth` command. There are two arguments to this command: the first contains the journal name information and the second contains the author names and paper title.

```
\markboth{Journal of Quantum Electronics,
Vol. 1, No. 1, January 2021}
{Author1, Author2,
\MakeLowercase{\textit{(et al.)}:
Paper Title}
```

D. Copyright Line

For Transactions and Journals papers, this is not necessary to use at the submission stage of your paper. The IEEE production process will add the appropriate copyright line. If you are writing a conference paper, please see the "IEEEtran_HOWTO.pdf" for specific information on how to code "Publication ID Marks".

E. Abstracts

The abstract is the first element of a paper after the `\maketitle` macro is invoked. The coding is simply:

```
\begin{abstract}
Text of your abstract.
\end{abstract}
```

Please try to avoid mathematical and chemical formulas in the abstract.

F. Index Terms

The index terms are used to help other researchers discover your paper. Each society may have it's own keyword set. Contact the EIC of your intended publication for this list.

```
\begin{IEEEkeywords}
Broad band networks, quality of service
\end{IEEEkeywords}
```

VIII. How to Create Common Body Elements

The following sections describe common body text elements and how to code them.

A. Initial Drop Cap Letter

The first text paragraph uses a "drop cap" followed by the first word in ALL CAPS. This is accomplished by using the `\IEEEPARstart` command as follows:

```
\IEEEPARstart{T}{his} is the first paragraph
of your paper. . .
```

B. Sections and Subsections

Section headings use standard LaTeX commands: `\section`, `\subsection` and `\subsubsection`. Numbering is handled automatically for you and varies according to type of publication. It is common to not indent the first paragraph following a section head by using `\noindent` as follows:

```
\section{Section Head}
\noindent The text of your paragraph . . .
```

C. Citations to the Bibliography

The coding for the citations are made with the LaTeX `\cite` command. This will produce individual bracketed reference numbers in the IEEE style. At the top of your LaTeX file you should include:

```
\usepackage{cite}
```

For a single citation code as follows:

```
see \cite{ams}
```

This will display as: see [1]

For multiple citations code as follows:

```
\cite{ams,oxford,lacomp}
```

This will display as [1], [2], [3]

D. Figures

Figures are coded with the standard LaTeX commands as follows:

```
\begin{figure}[!t]
\centering
\includegraphics[width=2.5in]{fig1}
\caption{This is the caption for one fig.}
\label{fig1}
\end{figure}
```

The [!t] argument enables floats to the top of the page to follow IEEE style. Make sure you include:

```
\usepackage{graphicx}
```

at the top of your LaTeXfile with the other package declarations.

To cross-reference your figures in the text use the following code example:

```
See figure \ref{fig1} ...
```

This will produce:
See figure 1 . . .

Fig. 1. This is the caption for one fig.

TABLE I
A SIMPLE TABLE EXAMPLE.

Order of filter	Arbitrary coefficients e_m	coefficients b_{ij}
1	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$
2	$\beta_{22} = (~1, -1, -1, ~~1, ~~1, ~~1)$	
3	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$,

E. Tables

Tables should be coded with the standard LaTeX coding. The following example shows a simple table.

```
\begin{table}
\begin{center}
\caption{Filter design equations  ...}
\label{tab1}
\begin{tabular}{| c | c | c |}
\hline
Order & Arbitrary coefficients &
coefficients\\
of filter & $e_m$ &   $b_{ij}$ \\
\hline
1& $b_{ij}=\hat{e}.\hat{\beta_{ij}}$,
& $b_{00}=0$\\
\hline
2&$\beta_{22}=(~1,-1,-1,~~1,~~1,~~1)$ &\\
\hline
3& $b_{ij}=\hat{e}.\hat{\beta_{ij}}$,
& $b_{00}=0$,\\
\hline
\end{tabular}
\end{center}
\end{table}
```

To reference the table in the text, code as follows:

```
Table~\ref{tab1} lists the closed-form...
```

to produce:

Table I lists the closed-form . . .

F. Lists

In this section, we will consider three types of lists: simple unnumbered, numbered and bulleted. There have been numerous options added to IEEEtran to enhance the creation of lists. If your lists are more complex than those shown below, please refer to the "IEEEtran_HOWTO.pdf" for additional options.

A plain unnumbered list

bare_jrnl.tex
bare_conf.tex
bare_jrnl_compsoc.tex
bare_conf_compsoc.tex
bare_jrnl_comsoc.tex

coded as:

```
\begin{list}{}{}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{list}
```

A simple numbered list

1) bare_jrnl.tex
2) bare_conf.tex
3) bare_jrnl_compsoc.tex
4) bare_conf_compsoc.tex
5) bare_jrnl_comsoc.tex

coded as:

```
\begin{enumerate}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{enumerate}
```

A simple bulleted list

- bare_jrnl.tex
- bare_conf.tex
- bare_jrnl_compsoc.tex
- bare_conf_compsoc.tex
- bare_jrnl_comsoc.tex

coded as:

```
\begin{itemize}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{itemize}
```

G. Other Elements

For other less common elements such as Algorithms, Theorems and Proofs, and Floating Structures such as pagewide tables, figures or equations, please refer to the "IEEEtran_HOWTO.pdf" section on "Double Column Floats."

IX. How to Create Common Back Matter Elements

The following sections demonstrate common back matter elements such as Acknowledgments, Bibliographies, Appendicies and Author Biographies.

A. Acknowledgments

This should be a simple paragraph before the bibliography to thank those individuals and institutions who have supported your work on this article.

```
\section{Acknowledgments}
\noindent Text describing those who
supported your paper.
```

B. Bibliographies

References Simplified: A simple way of composing references is to use the `\bibitem` macro to define the beginning of a reference as in the following examples:

[6] H. Sira-Ramirez. "On the sliding mode control of nonlinear systems," *Systems & Control Letters*, vol. 19, pp. 303–312, 1992.
coded as:

```
\bibitem{Sira3}
H. Sira-Ramirez. ``On the sliding mode
control of nonlinear systems,''
\textit{Systems \& Control Letters},
vol. 19, pp. 303--312, 1992.
```

[7] A. Levant."Exact differentiation of signals with unbounded higher derivatives," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp. 5585–5590, 2006.
coded as:

```
\bibitem{Levant}
A. Levant. ``Exact differentiation of
signals with unbounded higher
derivatives,''  in \textit{Proceedings
of the 45th IEEE Conference on
Decision and Control}, San Diego,
California, USA, pp. 5585--5590, 2006.
```

[8] M. Fliess, C. Join, and H. Sira-Ramirez. "Non-linear estimation is easy," *International Journal of Modelling, Identification and Control*, vol. 4, no. 1, pp. 12–27, 2008.
coded as:

```
\bibitem{Cedric}
M. Fliess, C. Join, and H. Sira-Ramirez.
``Non-linear estimation is easy,''
\textit{International Journal of Modelling,
Identification and Control}, vol. 4,
no. 1, pp. 12--27, 2008.
```

[9] R. Ortega, A. Astolfi, G. Bastin, and H. Rodriguez. "Stabilization of food-chain systems using a port-controlled Hamiltonian description," in *Proceedings of the American Control Conference*, Chicago, Illinois, USA, pp. 2245–2249, 2000.
coded as:

```
\bibitem{Ortega}
R. Ortega, A. Astolfi, G. Bastin, and H.
Rodriguez. ``Stabilization of food-chain
systems using a port-controlled Hamiltonian
description,'' in \textit{Proceedings of the
American Control Conference}, Chicago,
Illinois, USA, pp. 2245--2249, 2000.
```

C. Accented Characters in References

When using accented characters in references, please use the standard LaTeX coding for accents. **Do not use math coding for character accents**. For example:

```
\'e, \"o, \`a, \~e
```

will produce: é, ö, à, ẽ

D. Use of BibTeX

If you wish to use BibTeX, please see the documentation that accompanies the IEEEtran Bibliography package.

E. Biographies and Author Photos

Authors may have options to include their photo or not. Photos should be a bit-map graphic (.tif or .jpg) and sized to fit in the space allowed. Please see the coding samples below:

```
\begin{IEEEbiographynophoto}{Jane Doe}
Biography text here without a photo.
\end{IEEEbiographynophoto}
```

or a biography with a photo

```
\begin{IEEEbiography}[{\includegraphics
[width=1in,height=1.25in,clip,
keepaspectratio]{fig1.png}}]
{IEEE Publications Technology Team}
In this paragraph you can place
your educational, professional background
and research and other interests.
\end{IEEEbiography}
```

Please see the end of this document to see the output of these coding examples.

X. Mathematical Typography and Why It Matters

Typographical conventions for mathematical formulas have been developed to **provide uniformity and clarity of presentation across mathematical texts**. This enables the readers of those texts to both understand the author's ideas and to grasp new concepts quickly. While software such as LATEX and MathType® can produce aesthetically pleasing math when used properly, it is also very easy to misuse the software, potentially resulting in incorrect math display.

IEEE aims to provide authors with the proper guidance on mathematical typesetting style and assist them in writing the best possible article.

As such, IEEE has assembled a set of examples of good and bad mathematical typesetting. You will see how various issues are dealt with. The following publications have been referenced in preparing this material:

Mathematics into Type, published by the American Mathematical Society
The Printing of Mathematics, published by Oxford University Press
The LaTeXCompanion, by F. Mittelbach and M. Goossens
More Math into LaTeX, by G. Grätzer
AMS-StyleGuide-online.pdf, published by the American Mathematical Society

Further examples can be seen at http://journals.ieeeauthorcenter.ieee.org/wp-content/uploads/sites/7/IEEE-Math-Typesetting-Guide.pdf

A. Display Equations

A simple display equation example shown below uses the “equation” environment. To number the equations, use the `\label` macro to create an identifier for the equation. LaTeX will automatically number the equation for you.

$$x = \sum_{i=0}^{n} 2{i} Q. \tag{1}$$

is coded as follows:

```
\begin{equation}
\label{deqn_ex1}
x = \sum_{i=0}^{n} 2{i} Q.
\end{equation}
```

To reference this equation in the text use the `\ref` macro. Please see (1)
is coded as follows:

```
Please see (\ref{deqn_ex1})
```

B. Equation Numbering

Consecutive Numbering: Equations within an article are numbered consecutively from the beginning of the article to the end, i.e., (1), (2), (3), (4), (5), etc. Do not use roman numerals or section numbers for equation numbering.

Appendix Equations: The continuation of consecutively numbered equations is best in the Appendix, but numbering as (A1), (A2), etc., is permissible.

Hyphens and Periods: Hyphens and periods should not be used in equation numbers, i.e., use (1a) rather than (1-a) and (2a) rather than (2.a) for sub-equations. This should be consistent throughout the article.

C. Multi-line equations and alignment

Here we show several examples of multi-line equations and proper alignments.

A single equation that must break over multiple lines due to length with no specific alignment.

$$\begin{multline}\text{The first line of this example}\\ \text{The second line of this example}\\ \text{The third line of this example} \tag{2}\end{multline}$$

is coded as:

```
\begin{multline}
\text{The first line of this example}\\
\text{The second line of this example}\\
\text{The third line of this example}
\end{multline}
```

A single equation with multiple lines aligned at the = signs

$$\begin{align} a &= c+d \tag{3}\\ b &= e+f \tag{4}\end{align}$$

is coded as:

```
\begin{align}
a &= c+d \\
b &= e+f
\end{align}
```

The `align` environment can align on multiple points as shown in the following example:

$$\begin{align} x &= y & X &=Y & a &=bc \tag{5}\\ x' &= y' & X' &=Y' & a' &=bz \tag{6}\end{align}$$

is coded as:

```
\begin{align}
x &= y & X & =Y & a &=bc\\
x' &= y' & X' &=Y' &a' &=bz
\end{align}
```

D. Subnumbering

The amsmath package provides a `subequations` environment to facilitate subnumbering. An example:

$$\begin{align} f&=g \tag{7a}\\ f' &=g' \tag{7b}\\ \mathcal{L}f &= \mathcal{L}g \tag{7c}\end{align}$$

is coded as:

```
\begin{subequations}\label{eq:2}
\begin{align}
f&=g \label{eq:2A}\\
f' &=g' \label{eq:2B}\\
\mathcal{L}f &= \mathcal{L}g \label{eq:2c}
\end{align}
\end{subequations}
```

E. Matrices

There are several useful matrix environments that can save you some keystrokes. See the example coding below and the output.

A simple matrix:

$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \tag{8}$$

is coded as:

```
\begin{equation}
\begin{matrix}  0 &  1 \\
1 &  0 \end{matrix}
\end{equation}
```

A matrix with parenthesis

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{9}$$

is coded as:

```
\begin{equation}
\begin{pmatrix} 0 & -i \\
 i &  0 \end{pmatrix}
\end{equation}
```

A matrix with square brackets

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{10}$$

is coded as:

```
\begin{equation}
\begin{bmatrix} 0 & -1 \\
1 &  0 \end{bmatrix}
\end{equation}
```

A matrix with curly braces

$$\begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix} \tag{11}$$

is coded as:

```
\begin{equation}
\begin{Bmatrix} 1 &  0 \\
0 & -1 \end{Bmatrix}
\end{equation}
```

A matrix with single verticals

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \tag{12}$$

is coded as:

```
\begin{equation}
\begin{vmatrix} a &  b \\
c &  d \end{vmatrix}
\end{equation}
```

A matrix with double verticals

$$\begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix} \tag{13}$$

is coded as:

```
\begin{equation}
\begin{Vmatrix} i &  0 \\
0 & -i \end{Vmatrix}
\end{equation}
```

F. Arrays

The `array` environment allows you some options for matrix-like equations. You will have to manually key the fences, but you'll have options for alignment of the columns and for setting horizontal and vertical rules. The argument to `array` controls alignment and placement of vertical rules.

A simple array

$$\left(\begin{array}{cccc} a+b+c & uv & x-y & 27\\ a+b & u+v & z & 134 \end{array} \right) \tag{14}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccc}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

A slight variation on this to better align the numbers in the last column

$$\left(\begin{array}{cccr} a+b+c & uv & x-y & 27\\ a+b & u+v & z & 134 \end{array} \right) \tag{15}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccr}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

An array with vertical and horizontal rules

$$\left(\begin{array}{c|c|c|r} a+b+c & uv & x-y & 27\\ \hline a+b & u+v & z & 134 \end{array} \right) \tag{16}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{c|c|c|r}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

Note the argument now has the pipe "|" included to indicate the placement of the vertical rules.

G. Cases Structures

Many times we find cases coded using the wrong environment, i.e., `array`. Using the `cases` environment will save keystrokes (from not having to type the `\left\lbrace`) and automatically provide the correct column alignment.

$$z_m(t) = \begin{cases} 1, & \text{if } \beta_m(t) \\ 0, & \text{otherwise.} \end{cases}$$

is coded as follows:

```
\begin{equation*}
{z_m(t)} =
\begin{cases}
1,&{\text{if}}\ {\beta }_m(t),\\
{0,}&{\text{otherwise.}}
\end{cases}
\end{equation*}
```

Note that the "&" is used to mark the tabular alignment. This is important to get proper column alignment. Do not use `\quad` or other fixed spaces to try and align the columns. Also, note the use of the `\text` macro for text elements such as "if" and "otherwise".

H. Function Formatting in Equations

In many cases there is an easy way to properly format most common functions. Use of the \ in front of the function name will in most cases, provide the correct formatting. When this does not work, the following example provides a solution using the `\text` macro.

$$d_{R}^{KM} = \underset{d_{l}^{KM}}{\text{arg min}} \{ d_{1}^{KM}, \ldots, d_{6}^{KM}\}.$$

is coded as follows:

```
\begin{equation*}
 d_{R}^{KM} = \underset {d_{l}^{KM}}
 {\text{arg min}} \{ d_{1}^{KM},
 \ldots,d_{6}^{KM}\}.
\end{equation*}
```

I. Text Acronyms inside equations

This example shows where the acronym "MSE" is coded using `\text{}` to match how it appears in the text.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_{i} - \hat{Y_{i}})^{2}$$

```
\begin{equation*}
 \text{MSE} = \frac {1}{n}\sum _{i=1}^{n}
(Y_{i} - \hat {Y_{i}})^{2}
\end{equation*}
```

J. Obsolete Coding

Avoid the use of outdated environments, such as `eqnarray` and $$ math delimiters, for display equations. The $$ display math delimiters are left over from PlainTeX and should not be used in LATEX, ever. Poor vertical spacing will result.

K. Use Appropriate Delimiters for Display Equations

Some improper mathematical coding advice has been given in various YouTube™ videos on how to write scholarly articles, so please follow these good examples:

For **single-line unnumbered display equations**, please use the following delimiters:

```
\[ . . . \] or
```

```
\begin{equation*} . . . \end{equation*}
```

Note that the * in the environment name turns off equation numbering.

For **multiline unnumbered display equations** that have alignment requirements, please use the following delimiters:

```
\begin{align*} . . . \end{align*}
```

For **single-line numbered display equations**, please use the following delimiters:

```
\begin{equation} . . . \end{equation}
```

For **multiline numbered display equations**, please use the following delimiters:

```
\begin{align} . . . \end{align}
```

XI. LaTeX Package Suggestions

Immediately after your documenttype declaration at the top of your LATEX file is the place where you should declare any packages that are being used. The following packages were used in the production of this document.

```
\usepackage{amsmath,amsfonts}
\usepackage{algorithmic}
\usepackage{array}
\usepackage[caption=false,font=normalsize,
   labelfont=sf,textfont=sf]{subfig}
\u00sepackage{textcomp}
\usepackage{stfloats}
\usepackage{url}
\usepackage{verbatim}
\usepackage{graphicx}
\usepackage{balance}
```

XII. Additional Advice

Please use "soft" (e.g., `\eqref{Eq}`) or `(\ref{Eq})` cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please note that the `{subequations}` environment in LATEX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Please do not use `\nonumber` or `\notag` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

XIII. A Final Checklist

1) Make sure that your equations are numbered sequentially and there are no equation numbers missing or duplicated. Avoid hyphens and periods in your equation numbering. Stay with IEEE style, i.e., (1), (2), (3) or for sub-equations (1a), (1b). For equations in the appendix (A1), (A2), etc..
2) Are your equations properly formatted? Text, functions, alignment points in cases and arrays, etc.
3) Make sure all graphics are included.
4) Make sure your references are included either in your main LaTeX file or a separate .bib file if calling the external file.

References

[1] *Mathematics into Type*, American Mathematical Society. Online available:
[2] T.W. Chaundy, P.R. Barrett and C. Batey, *The Printing of Mathematics*, Oxford University Press. London, 1954.
[3] *The LaTeXCompanion*, by F. Mittelbach and M. Goossens
[4] *More Math into LaTeX*, by G. Grätzer
[5] *AMS-StyleGuide-online.pdf,* published by the American Mathematical Society
[6] H. Sira-Ramirez. "On the sliding mode control of nonlinear systems," *Systems & Control Letters*, vol. 19, pp. 303–312, 1992.
[7] A. Levant. "Exact differentiation of signals with unbounded higher derivatives," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp. 5585–5590, 2006.
[8] M. Fliess, C. Join, and H. Sira-Ramirez. "Non-linear estimation is easy," *International Journal of Modelling, Identification and Control*, vol. 4, no. 1, pp. 12–27, 2008.
[9] R. Ortega, A. Astolfi, G. Bastin, and H. Rodriguez. "Stabilization of food-chain systems using a port-controlled Hamiltonian description," in *Proceedings of the American Control Conference*, Chicago, Illinois, USA, pp. 2245–2249, 2000.

Jane Doe Biography text here without a photo.

IEEE Publications Technology Team In this paragraph you can place your educational, professional background and research and other interests.