A Guide to *TensorShop*

tensorshop.github.io

Jonah Stockwell and Josh Alman

Contents

I. Introduction

Matrix multiplication is a fundamental algebraic operation with diverse practical applications. As matrices get larger, the time it takes to multiply them increases dramatically. This time complexity grows in proportion to n^{ω} where n is the size of the n x n matrices being multiplied. Reducing the size of this exponent of matrix multiplication, ω, is thus a key goal. To reduce the time complexity, tensors are analyzed and manipulated in order to bound their 'value,' in turn bounding the exponent of matrix multiplication. Given that analysis and manipulation of tensors is cumbersome, a tool for visualizing, performing, and organizing tensor analysis is needed.

TensorShop provides a suite of tools for importing tensors, exporting two-dimensional and three-dimensional representations, exporting tensors for analysis in other mathematical software, easily generating common tensors, building new tensors, manipulating tensor axes, permuting indices, zooming in and out on tensors, panning around tensors, partitioning tensors, finding matrix multiplication subtensors, and performing operations such as zeroing out, monomial degeneration by labels, solving monomial degeneration by linear programming, and Kronecker products. *TensorShop* is built for the browser using vanilla Javascript, the three.js library, and the Flask Python microframework. *TensorShop* therefore enables easy visualization and analysis of tensors, facilitating breakthroughs in the reduction of the time complexity of matrix multiplication algorithms.

TensorShop is a tool for visualizing and manipulating "free" tensors with coefficients that are either 0 or 1. A tensor is called free if any two (i1, j1, k1), (i2, j2, k2) in the support differ in at least two entries [CGLVW21]. Historically, almost all the tensors studied in designing fast matrix multiplication algorithms have been free tensors. Additionally, free tensors are much easier to visualize, especially in three dimensions, than other tensors.

The operations supported by *TensorShop* are common operations from prior work from the matrix multiplication literature. We assume familiarity with them here, but we provide references about how each is typically used for the unfamiliar reader. We recommend [Blä13] as an introduction for any readers unacquainted with the basics of tensors and their relationship with matrix multiplication algorithms.

TensorShop includes a built-in tutorial that explains most of *TensorShop's* features. However, this guide contains more details about all of these features. It is recommended to first go through the tutorial, then read this guide for clarification.

II. The Workspace

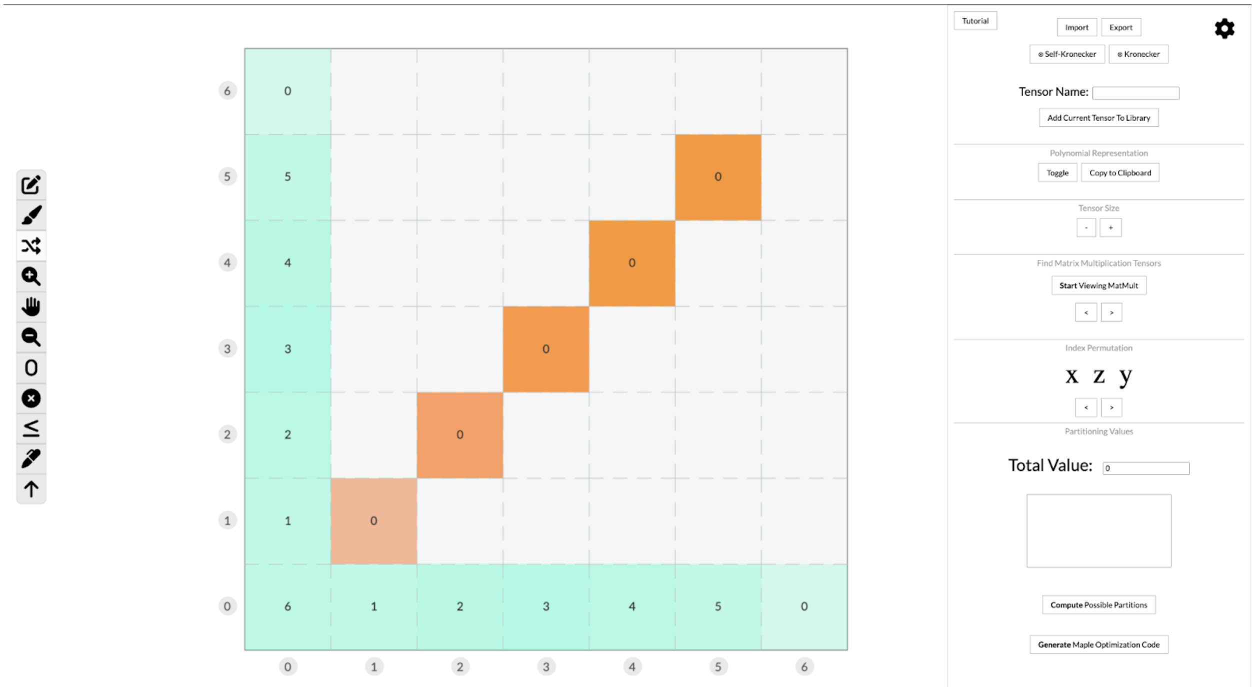


Figure 2.1. *The TensorShop GUI. The tools are on the left. The visualized "matrix form" of the tensor is in the middle, and the control panel is on the right.*

Matrix Form

In *TensorShop,* the user operates with the "matrix form" of the current loaded tensor (**Figure 2.2**). The figure depicts a CW-5 tensor, which is defined as follows:

$$x_6y_0z_0 + x_0y_6z_0 + x_0y_0z_6 + \sum_{i}^{5}(x_0y_iz_i + x_iy_0z_i + x_iy_iz_0)$$

In the matrix, each column corresponds to an x-variable, each row corresponds to a y-variable, and each cell's value corresponds to a z-variable. Each filled cell corresponds to a term in the **polynomial form** of the tensor.[1] Depending on the current context, annotations will appear around and over the matrix form. The colors do not change the definition of the tensor, they are simply a visual aid and can be changed in settings. See Section VIII for an explanation of the coloring of the matrix.

[1] Note that because we are working with free tensors, there will be at most one z variable in each cell.

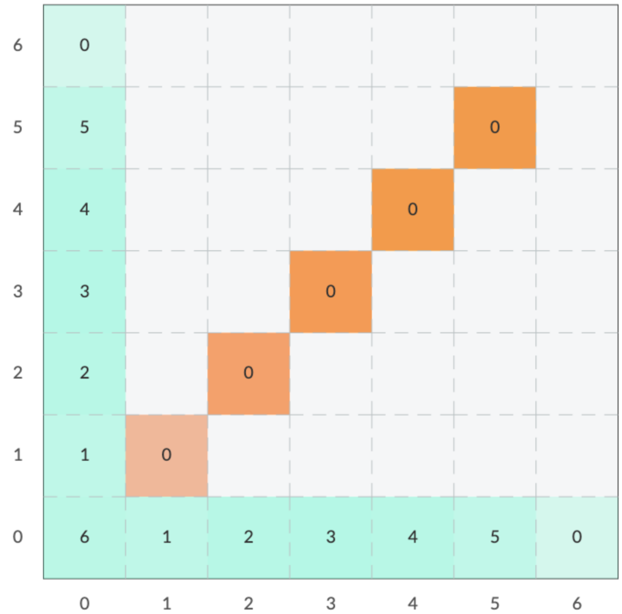


Figure 2.2. The "matrix form" of a Coppersmith-Winograd 5 tensor.

Polynomial Form

$$x_0y_0z_6 + x_1y_0z_1 + x_2y_0z_2 + x_3y_0z_3 + x_4y_0z_4 + x_5y_0z_5 + x_6y_0z_0 + x_0y_1z_1 + x_1y_1z_0 + x_0y_5z_5 + x_5y_5z_0 + x_0y_2z_2 + x_2y_2z_0 + x_0y_4z_4 + x_4y_4z_0 + x_0y_3z_3 + x_3y_3z_0 + x_0y_6z_0$$

Figure 2.3. The "polynomial form" of a Coppersmith-Winograd 5 tensor.

TensorShop can also display the "polynomial form" of the matrix (**Figure 2.3**). The polynomial representation can be toggled by the "Toggle" button in the "Polynomial Representation" section of the control panel (**Figure 2.4**). The representation can also be copied via the "Copy to Clipboard" button.

The Control Panel

The control panel (**Figure 2.1, right**) houses a variety of non-tool actions. Each of these actions are grouped in sections: Importing and Exporting, Polynomial Representation, Tensor Size, Find Matrix Multiplication, Index Permutation, and Partitioning Values. Some sections appear and disappear depending on the current context.

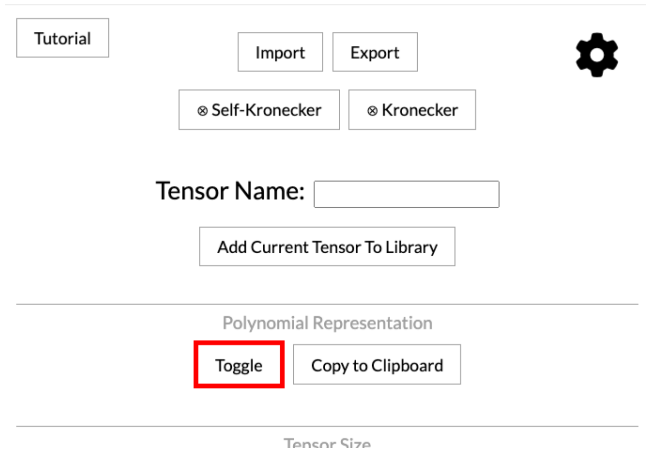


Figure 2.4. The control panel toggle button that shows or hides the "polynomial form" of the current tensor.

Settings

TensorShop also supports degrees of customization through the settings panel. To open the settings panel, click the cog icon in the top right corner of the control panel (**Figure 2.5**).

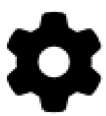


Figure 2.5. The cog icon to open the settings panel.

Settings Reset

If *TensorShop* needs to be reset, simply enter the settings panel by clicking on the cog icon and click the "Hard Reset" button at the bottom of the panel.

URL Reset

Although unlikely, if *TensorShop* enters a state where the user cannot even navigate to the hard reset button in settings, simply append "?reset=true" to the url to fully reset *TensorShop*.

The Toolbar

Figure 2.6. *The toolbar; each icon corresponds to a selectable “tool.”*

The toolbar houses a variety of tools for editing, viewing, manipulating, and analyzing the tensor (**Figure 2.6**). Each icon corresponds to a different tool that can be selected. When hovering over a tool icon, a tooltip with the name of the tool will appear.

In this user manual, actions that are associated with a particular tool will have that tool icon next to the heading. For example, the section pertaining to zooming in with the zoom in tool (magnifying glass), would look like this:

Zoom In

Undoing and Redoing

TensorShop preserves a history of changes in the tensor, allowing the user to undo and redo changes. To undo a change, simply press the **z** key. To redo an undone change, simply press the **x** key. Redos can only be performed on an undone change before any conflicting changes are made.

III. Moving Around the Workspace

Zoom In

If a tensor starts getting too big, the user may want to zoom in (**Figure 3.1**). The “Zoom In” tool, represented by a plus sign inside of magnifying glass, facilitates this. Simply select the tool and click where on the matrix form you want to zoom into (**Figure 3.2**).

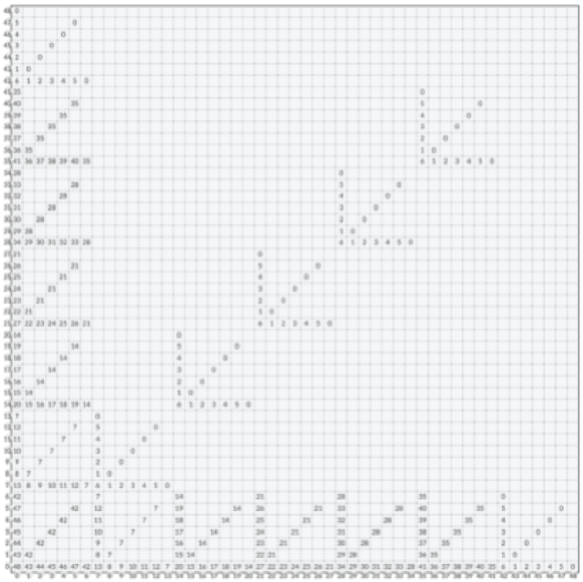


Figure 3.1. *A self Kronecker product of a CW-5 tensor.*

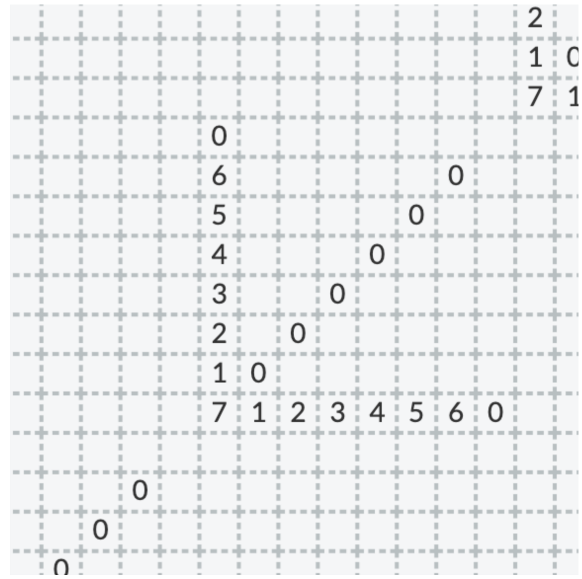


Figure 3.2. *A zoomed-in portion of a self Kronecker product of a CW-5 tensor.*

 Pan Around

The “Pan” tool, represented by the open hand, is used to move around the matrix form of the tensor. Select the tool and click and drag on the workspace to move around.

 Zoom Out

The “Zoom Out” tool, represented by the minus sign inside of a magnifying glass, is used to zoom back out when zoomed in. Select the tool and simply click anywhere to zoom out.

IV. Importing

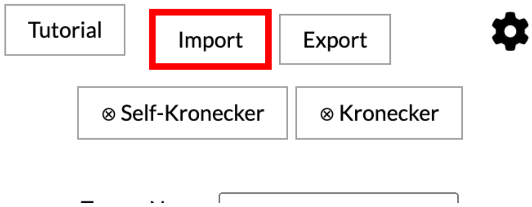


Figure 4.1. The import button in the control panel.

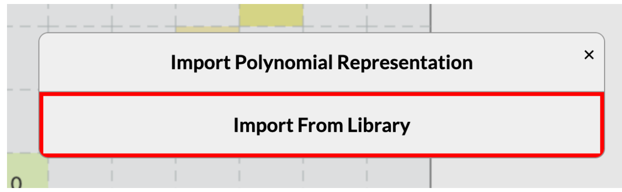


Figure 4.2. The "Import From Library" option in the import dialog.

Generating Common Tensors

There are a few common tensors that are built into *TensorShop*. They are housed in the **tensor library** (**Figure 4.3**). To access it, click the "Import" button (**Figure 4.1**), select "Import From Library" (**Figure 4.2**).

Tensors can then be imported with a click. For tensor generators, there are fields that control the resultant tensor. For example, the cyclic group generator generates cyclic group tensors with a particular operation mod a particular value of N.[2]

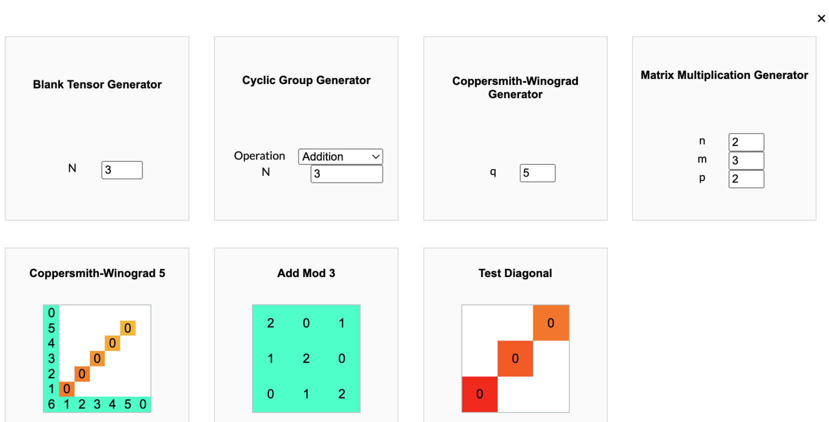


Figure 4.3. The tensor library.

Generating Blank Tensors

Although *TensorShop* provides a variety of "generators" for common tensors, the user may want to create their own tensors. To do so, open the **tensor library** and find the "Blank Tensor Generator," change N to the desired size and click.

[2]See [CU03] for more background on the use of group tensors.

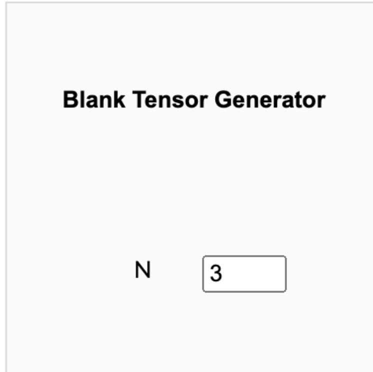


Figure 4.4. The blank tensor generator.

Importing Polynomial Representation

The user can import a polynomial representation such as the following example:
$x_0y_0z_0+x_1y_0z_1+x_2y_0z_2+x_1y_1z_2+x_2y_1z_0+x_0y_2z_2+x_2y_2z_1+x_0y_1z_1+x_1y_2z_0$

To do so, click the "Import" button (**Figure 4.1**), select "Import Polynomial Representation" (**Figure 4.5**). Then, paste the representation into the field and click import (**Figure 4.6**).

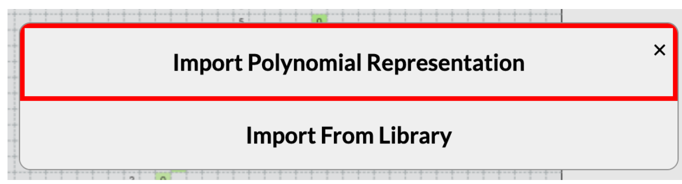
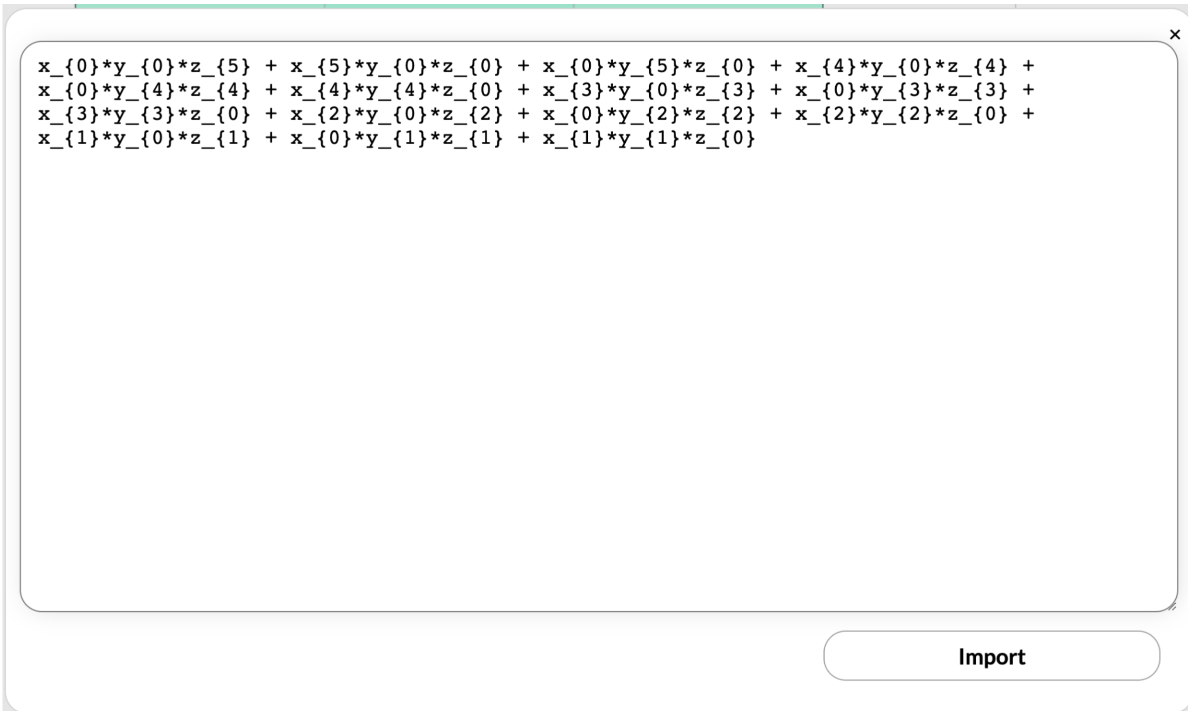


Figure 4.5. The "Import Polynomial Representation" button in the import button dialog.



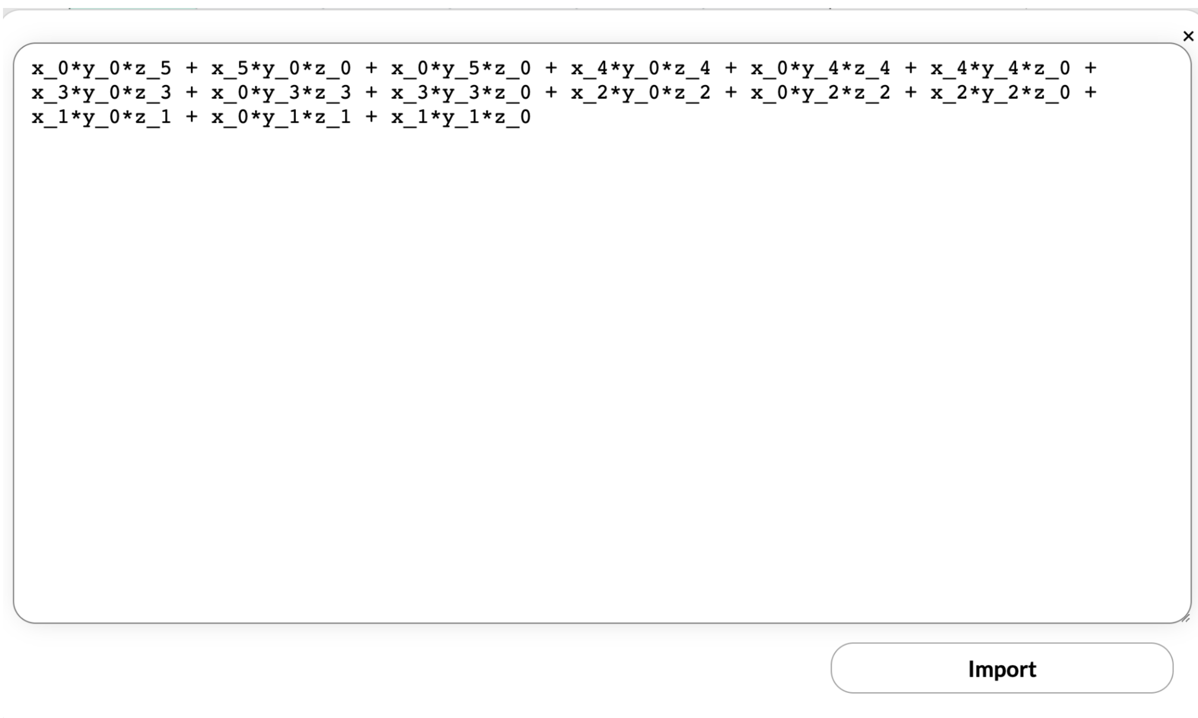


Figure 4.6. *The import polynomial representation window with example inputs for a CW-4 tensor.*

V. Building New Tensors

Single Cell Editing

To edit a single cell in the current tensor, use the "Edit Single Item" tool, represented by the pencil in a box icon. Then, simply click on a cell in the matrix form of the tensor and type in the new value.

Multiple Cell Painting

Common tensors tend to have sequential patterns. The "Paint Multiple Items" tool, represented by the paintbrush, allows the user to draw a path through multiple cells. Click on a cell and drag out a path. On release of the mouse, the path will be filled in by an expression (**Figure 5.1**). The expression is evaluated at each cell along the path in terms of *i*, the index along the path, and *s*, the size of the tensor. Expressions for the "Paint Multiple Items" tools should evaluate to integers and only use the operators: *+* (addition), - (subtraction), */* (division), *** (multiplication), *%* (modulo), and ^ (exponentiation).

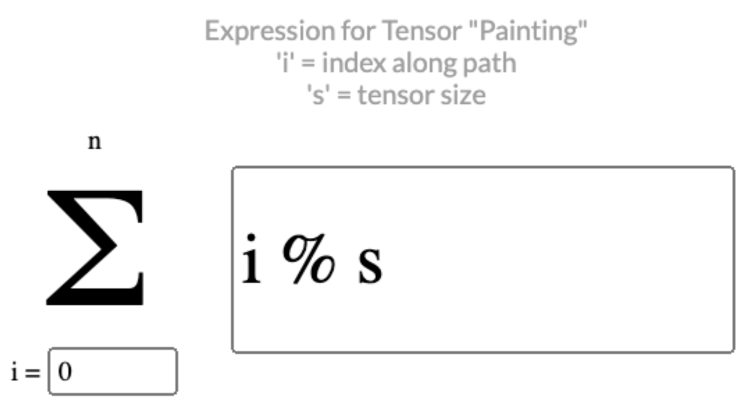


Figure 5.1. *The tensor "painting" section for an expression along a path.*

VI. Exporting

There are a variety of ways to export an image representation of the current tensor.

Exporting 2D Representations

To export a 2D representation of the current tensor, click the “Export” button (**Figure 6.1**) in the control panel, opening a selection prompt (**Figure 6.2**). Then, select the desired image to export. The “Export Annotated 2D Matrix” option is exactly the version of the matrix representation seen in the workspace. The “Export Colored 2D Matrix” option creates an image of the matrix without any annotations (only colors). The “Export Plain 2D Matrix” option simply creates an uncolored, unannotated image of the matrix.

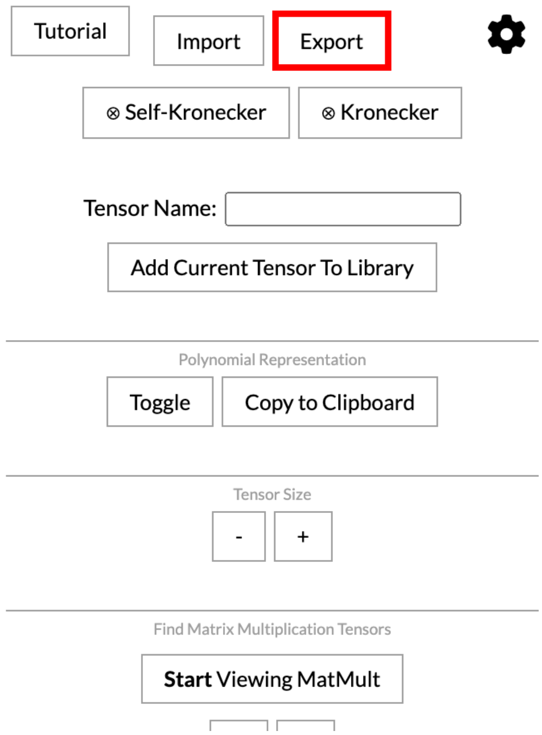


Figure 6.1. The “Export” button in the control panel.

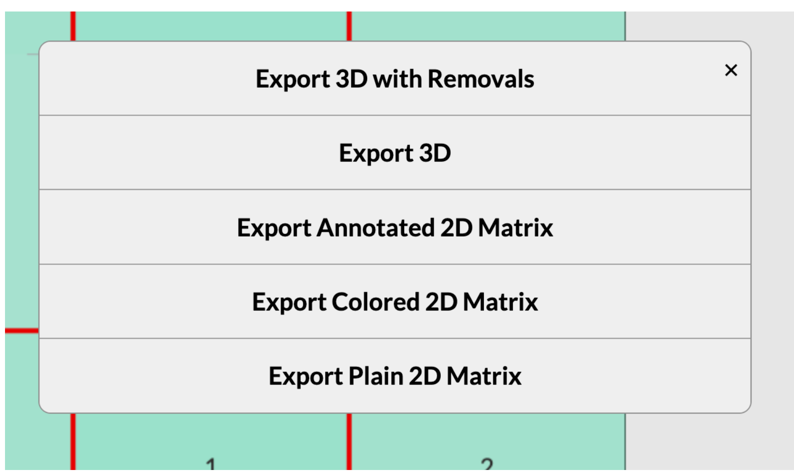


Figure 6.2. The export type selection dialog box that appears after clicking “Export” in the control panel.

Exporting 3D Representations

Exporting 3D representations is similar to exporting 2D representations. Just as with 2D representations, click the “Export” button in the control panel. Next, select one of the 3D options, opening the 3D viewer. Then, rotate around the model with the mouse and move around with the arrow keys. When the image is set up, click the “Export” button in the bottom right to generate the image version (**Figure 6.3**).

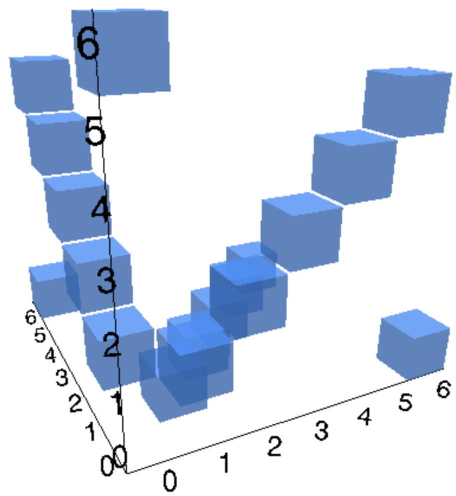


Figure 6.3. *A 3D representation of a CW-5 tensor exported from TensorShop.*

VII. Manipulating Tensor Axes

Swapping and Inserting Individual Rows and Columns

To swap individual rows and columns, select the “Swapping and Inserting” tool, represented by the crossing arrows. Then, click on a row or column index (**Figure 7.1**) and drag the row or column to its new location. If the row or column is on top of another row or column, they will be swapped. Otherwise, if the row or column is between two rows or columns, it will be inserted between.

Figure 7.1. *Row labels when the “Swapping and Inserting” tool is selected.*

Inserting an Interval of Rows and Columns

Inserting an interval of rows or columns (**Figure 7.2**) also uses the “Swapping and Inserting” tool. To select an interval, simply hold shift while clicking the desired beginning and end of the interval. Then, insert[3] as normal.

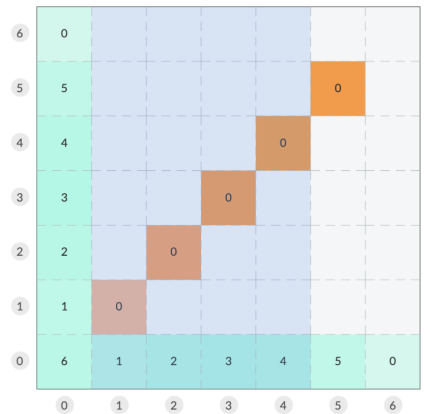


Figure 7.2. *An interval of columns selected on a CW-5 tensor.*

Index Permutation

Index permutation is a shuffling of the x, y, and z variables of a tensor. Index permutation is controlled in the “Index Permutation” section of the control panel. The top part of the section displays the current permutation, and the arrow buttons allow the user to permute.

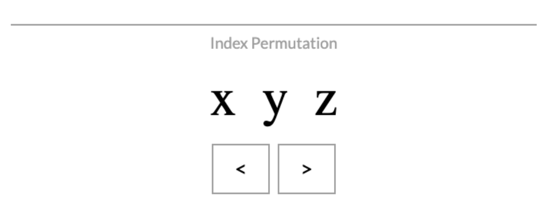


Figure 7.3. *The index permutation section in the control panel.*

[3] Presently, the user can only insert an interval of columns, not swap.

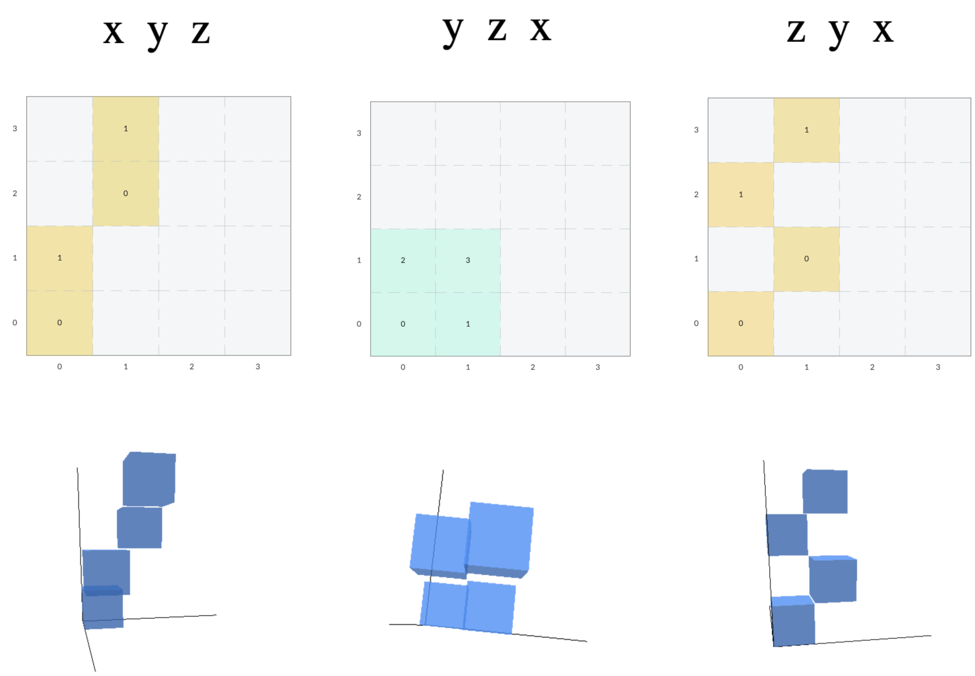


Figure 7.4. A ⟨1,2,2⟩ MM tensor under various index permutations.

VIII. Finding Matrix Multiplication Sub-Tensors

TensorShop finds "trivial" matrix multiplication sub-tensors for the current axis permutation. Graphically, these sub-tensors are represented by a "rectangle" filled with unique z-indices along a "diagonal" of n identical rectangles.

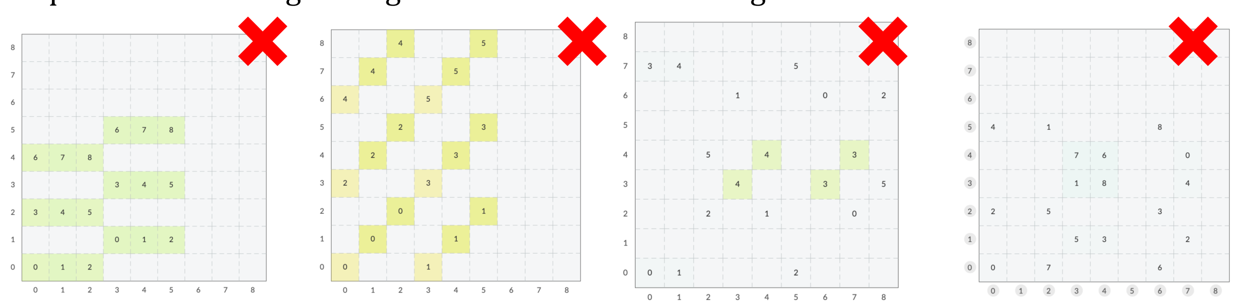


Figure 8.1. Examples of non-trivial matrix multiplication tensors: permutations of the matrix multiplication tensor ⟨3,3,2⟩.

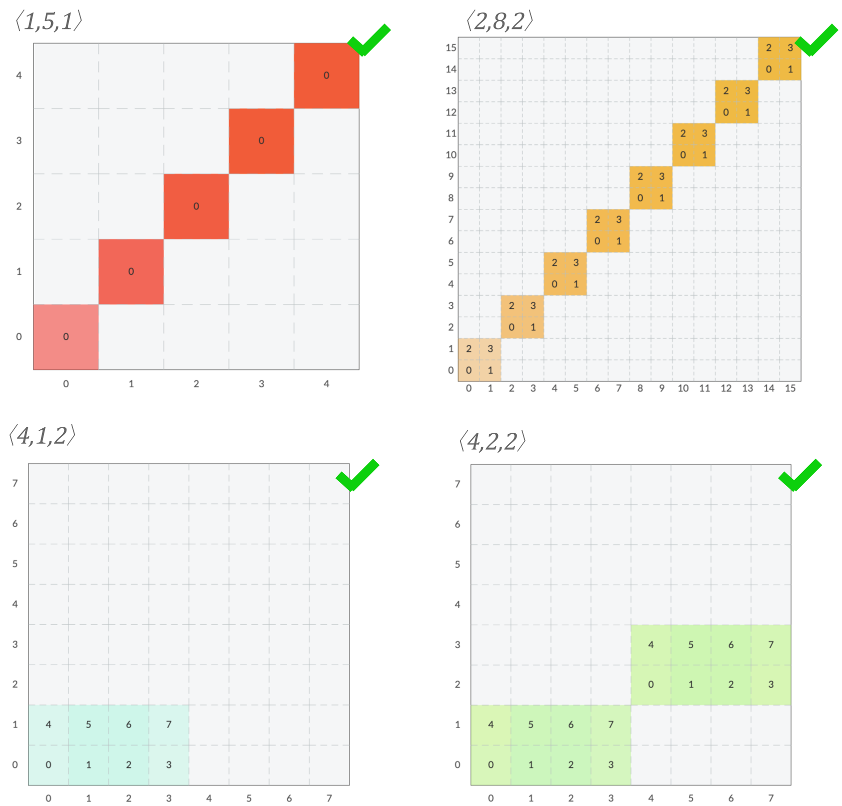


Figure 8.2. Examples of valid trivial matrix multiplication tensors.

View Matrix Multiplication Sub-Tensors

To see the found matrix multiplication tensors, click the “Start Viewing MatMult” button in the “Find Matrix Multiplication Tensors” section of the control panel and use the left and right arrow buttons to cycle through found tensors.

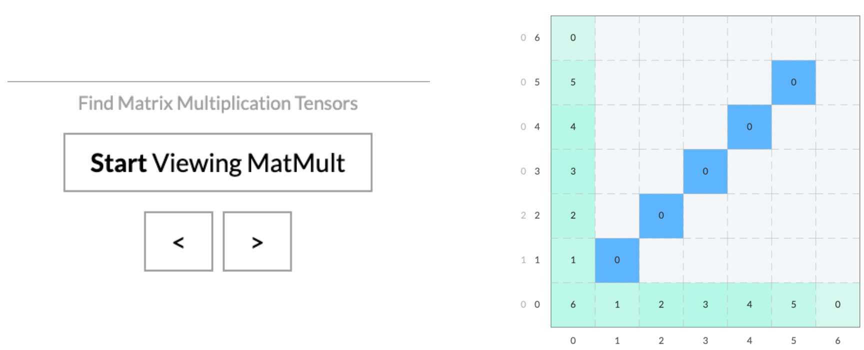


Figure 8.3. (Left) The “Find Matrix Multiplication Tensors” section of the control panel.

(Right) A CW-5 tensor with a MM sub-tensor highlighted.

Matrix Coloring

TensorShop colors each cell of the matrix with colors ranging from light green to red (by default) to highlight how many matrix multiplication sub-tensors the cell is part of, the length of the "diagonals" for sub-tensors, and the size of the "rectangles" of the sub-tensors. The coloring gradient can be modified in the menu.

IX. Zeroing Out[4]

0 Zeroing Out

See section 4.4 of [Alm19] or section 9.1 of [Blä13] for further background on zeroing out.

Zeroing out is the process of removing terms with a particular x, y, or z variable from a tensor. This corresponds to removing a particular column, row, or cell from the matrix form of the tensor. For example, zeroing out x_0 in a CW-2 tensor (**Figure 9.1**):

CW-2 tensor: $x_0y_0z_3 + x_1y_0z_1 + x_2y_0z_2 + x_3y_0z_0 + x_0y_1z_1 + x_1y_1z_0 + x_0y_3z_0$

***CW-2 tensor with** x_0* zeroed out*:* $x_1y_0z_1 + x_2y_0z_2 + x_3y_0z_0 + x_1y_1z_0 + x_2y_2z_0$

To zero out variables, use the "Zeroing Out" tool, represented by a "0" symbol. Next, click on a column index to zero out that x variable, row index to zero out that y variable, or cell to zero out that z variable. The removed terms will appear grayed out in the matrix-form of the tensor. Simply click on the grayed-out index to bring it back. Grayed-out terms are essentially removed and will not be used by other *TensorShop* tools and features, such as the polynomial representation or finding matrix multiplication sub-tensors.

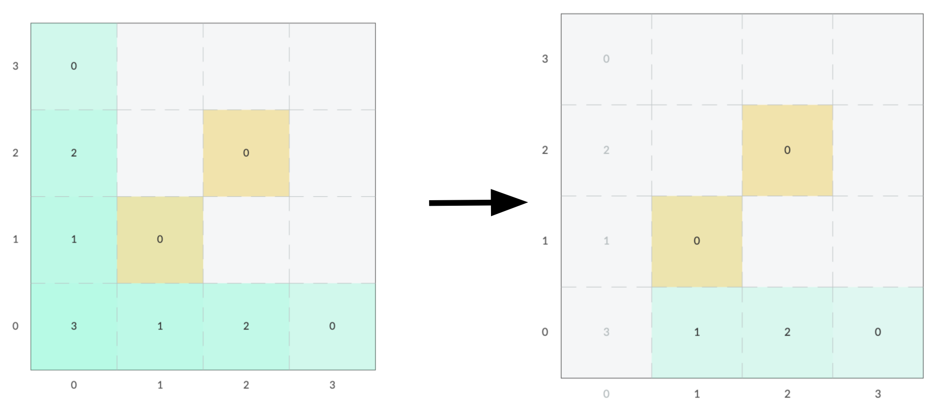


Figure 9.1. *A zeroing out of the first column (x_0) of a CW-2 tensor. The zeroed out cells and indices are displayed as gray.*

---

[4] Zeroing out is also sometimes referred to as combinatorial degeneration.

X. Monomial Degeneration

See section 4.4 of [Alm19] and section 9.1 of [Blä13].

The monomial degeneration algorithm goes as follows. First, assign numeric labels to the for each x and y index (row and column labels). Then, for each cell, compute the sum of the corresponding row and column labels. For each z-index, the cells with the minimal sum are kept, while all cells with non-minimal sums are removed. The minimal sum for each z-index is the z-label.

Monomial Degeneration by Labels

To perform the monomial degeneration operation by specifying x and y labels, use the "Monomial Degeneration" tool, represented by the 'x' inside a black circle. When the "Monomial Degeneration" tool is selected, label annotations will appear (**Figure 10.1**). To assign x and y labels, click on a label (found next to the index) and type in the desired new label. *TensorShop* will then compute the z-labels, putting the computed sums in the top right corner of every cell (**Figure 10.1**) and removing the terms of the same z variable with non-minimal sums. The z-labels are displayed in a table in the top-left corner of the workspace (**Figure 10.2**).

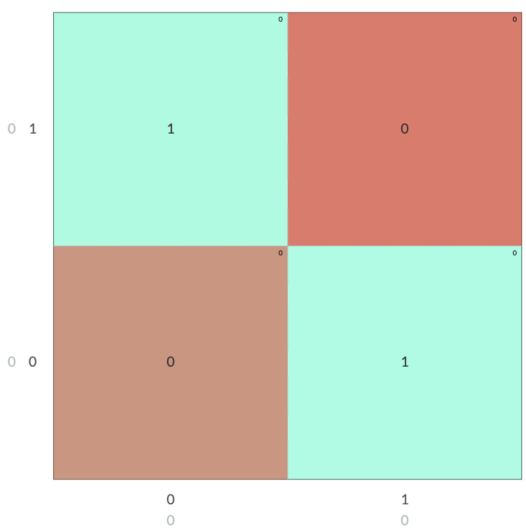


Figure 10.1. The matrix form of the tensor with monomial degeneration labels (next to the row and column labels) and computed sums (top right corner of each cell).

Z	Label
0	0
1	0

Figure 10.2. The table of z labels for monomial degeneration. Found in the top-left corner of the workspace.

≤ Solving Monomial Degeneration by Linear Programming

The “Solving Monomial Degeneration” tool, represented by the less than or equal sign, allows the user to determine whether there is a monomial degeneration that removes a particular subset of terms. To compute the best labels to remove terms, click on the cells to remove. Then, *TensorShop* will construct a linear optimization problem for the labels that will be solved by linear programming. Finally, *TensorShop* fills in the labels and performs the degeneration, removing the desired cells.

Solving Monomial Degeneration With Maple

If the user attempts to solve for monomial degeneration labels on a large tensor (>20 in an axis), *TensorShop* provides an option to, instead of computing the problem in-browser, export the linear optimization problem to maple (Figure 10.3).

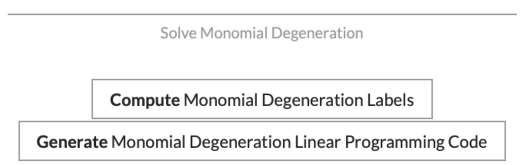


Figure 10.3. *Solve monomial degeneration options in the control panel for large tensors.*

XI. Kronecker Products

See section 5.2 of [Blä13] and Section 4.3 of [Alm19] for background on the Kronecker Product.

Given two tensors T_1 and T_2:

$$T_1 = \sum_{i=1}^{a_1}\sum_{j=1}^{b_1}\sum_{k=1}^{c_1} \alpha_{ijk} x_i y_j z_k \text{ (where } \alpha_{ijk} \text{ is 0 or 1)} \qquad T_2 = \sum_{i'=1}^{a_2}\sum_{j'=1}^{b_2}\sum_{k'=1}^{c_2} \beta_{i'j'k'} x_{i'} y_{j'} z_{k'} \text{ (where } \beta_{i'j'k'} \text{ is 0 or 1)}$$

The Kronecker product is:

$$T_1 \otimes T_2 = \sum_{i=1}^{a_1}\sum_{j=1}^{b_1}\sum_{k=1}^{c_1}\sum_{i'=1}^{a_2}\sum_{j'=1}^{b_2}\sum_{k'=1}^{c_2} \alpha_{ijk}\beta_{i'j'k'} x_{(i,i')} y_{(j,j')} z_{(k,k')}$$

Each term in the Kronecker product (⊗) corresponds to a pair of terms from a and b respectively. This is very similar to the notion of a tensor product. The Kronecker product is also sometimes signified by a square with a cross in it instead of a circle with a cross in it.

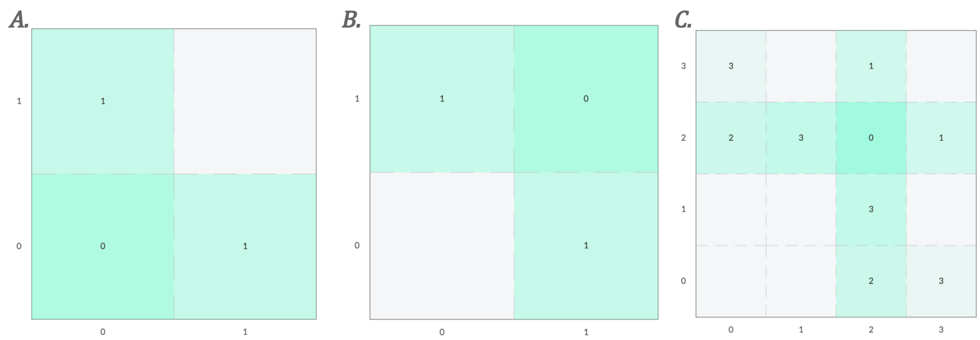


*Figure 11.1. A tensor **C** that is the kronecker product of the two tensors, **A** and **B**.*

To take a Kronecker product of two tensors **A** and **B** ($A \otimes B$), load tensor A in as the current tensor, then use the "Add Current Tensor to Library" button to add it to the tensor library (**Figure 11.2**). Then, load tensor B in as the current tensor. Finally, click the "Kronecker" button in the control panel and compute the Kronecker product of A and B (**Figure 11.3**). To take a self-Kronecker product (**Figure 11.4**), click the "Self-Kronecker" button in the control panel.

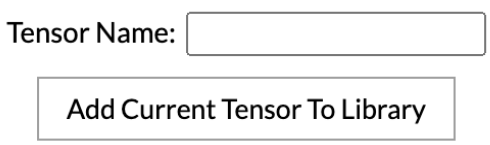


Figure 11.2. The "Add Current Tensor to Library" section in the control panel.

Figure 11.3. The compute Kronecker product in the control panel.

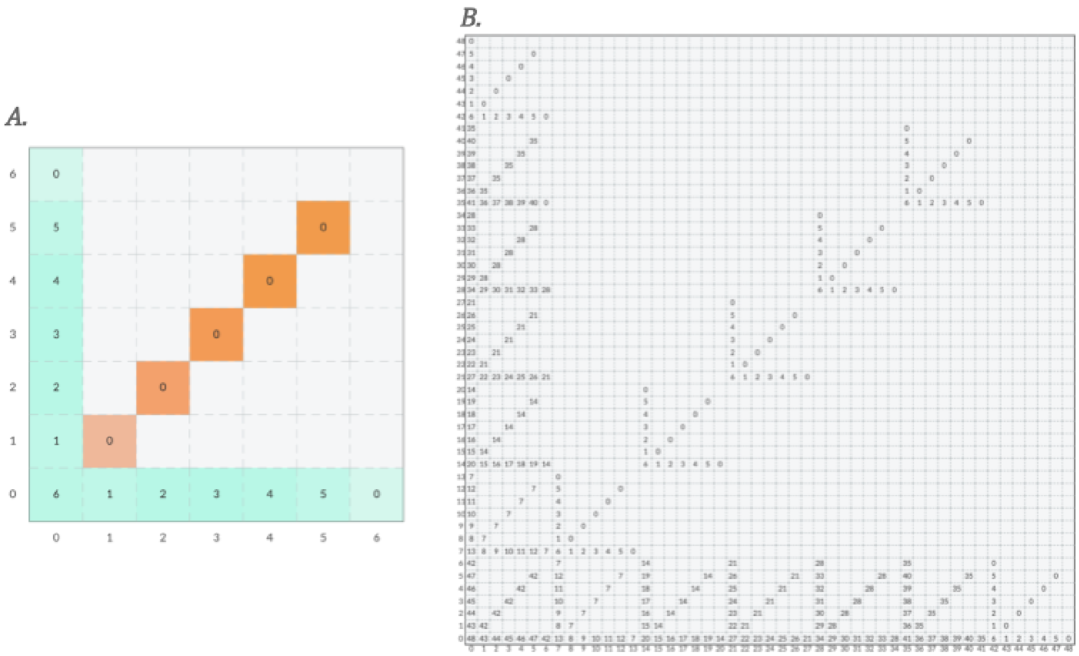


*Figure 11.4. A tensor **B** that is the self-Kronecker product of the CW-5 tensor **A**.*

Kronecker Product Labels

After performing a Kronecker product, *TensorShop* allows the user to toggle labels between normal indices and pairs of indices relative to their place in the original tensors. To do so, click the "Toggle ⊗ Labels" button that is now visible in the top of the control panel. Click the button again to toggle back. When toggled on, Kronecker labels can be edited with the Single Cell Editing tool, swapped around and inserted with the Swapping and Inserting tool,[5] and permuted with Index Permutation.

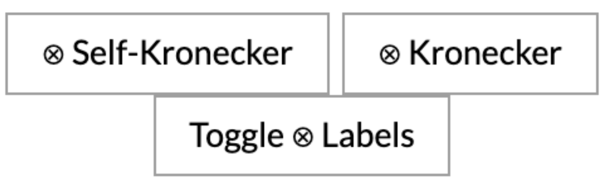


Figure 11.5. The 'Toggle Kronecker Labels' button located at the top of the control panel

[5] As opposed to with normal labels, row and column Kronecker labels stay with the rest of their row or column when swapped or inserted.

XII. Tensor Partitioning

In *TensorShop*, partitions are represented by vertical and horizontal lines between columns and rows. Visually, a valid partitioning scheme only shares indices along diagonals of partitions. Validly partitioned tensors are block tight tensors. We can only use tools like the laser method on tensors that are block tight tensors.

For more background on the usage of partitioning see Chapter 5 in [Alm19].

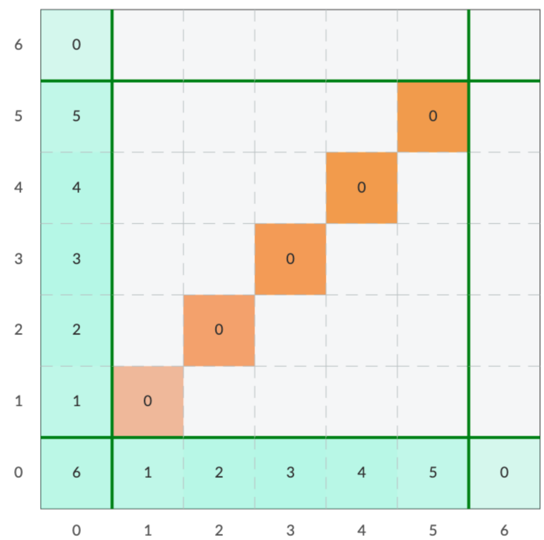


Figure 12.1. *A CW-5 tensor with a valid partitioning scheme.*

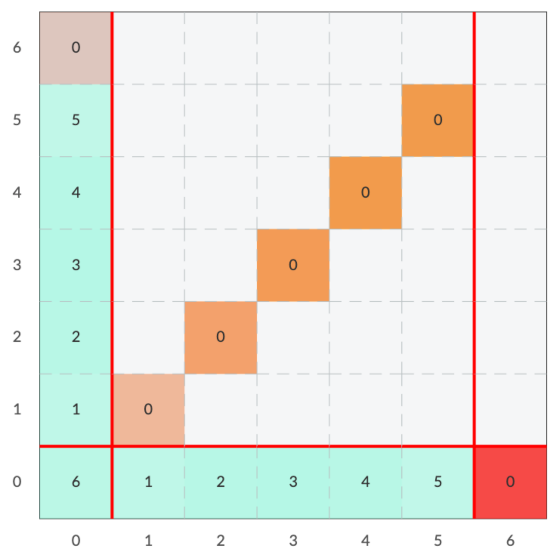


Figure 12.2. *A CW-5 tensor with an invalid partitioning scheme containing one index conflict.*

Drawing Partitions

To partition the matrix representation, select the “Partition Matrix” tool, represented by the pen icon. Then, click in between rows or columns to create a partition. If the partitioning scheme is valid, the partitions will appear green (**Figure 12.1**). If the partitioning scheme is invalid, the partitions will appear red and conflicting cells will appear highlighted red (**Figure 12.2**).

Zooming In and Out on Partitioned Regions

One may want to “zoom in” onto a partitioned region, making the region the current tensor and preserving the larger tensor in a “zoom stack”. To zoom in on a partition, simply select the “Partition Matrix” tool (pen), and **right click** on the desired partitioned region. To zoom back out, select the “Zoom Out on Partition” tool and click. If the user is using the “Partitioning Values” section of the control panel, the total value of the zoomed in region will fill in the corresponding cell in the value table for the partitioned tensor.

Exporting Maple Minimization Problem

TensorShop has the capacity to generate Maple code to maximize the “value” of the tensor as seen in the approach used in [CW87] and [Wil12]. This is of the form:

$$\frac{\prod_{i,j} V_{i,j}^{a_{i,j}}}{\left[\left(\sum_i a_i^{a_i}\right)\left(\sum_j b_j^{b_j}\right)\left(\sum_k c_k^{c_k}\right)\right]^{\frac{1}{3}}}$$

where

$a_{i,j}$ is a variable for each cell in the value table in the “Partitioning Values,”

a_i is the sum of a row $a_i = \sum_j a_{i,j}$

b_j is the sum of a column $b_j = \sum_i a_{i,j}$

c_k is the sum of a diagonal $c_k = \sum_{i=1}^{k} (a_{i,k-i+1})$

To copy the optimization code to the clipboard, click the “Generate Maple Optimization Code” button in the “Partitioning Values” section of the control panel (**Figure 12.3**). If there is no “Partitioning Values” section of the control panel, enable it in Settings (gear icon in top-right).

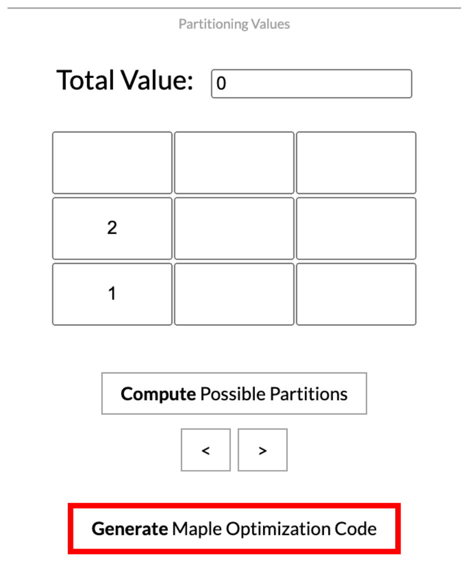


Figure 12.3. The "Generate Maple Optimization Code" button in the "Partitioning Values" section of the control panel.

XIII. Contact Us

We welcome feedback and suggestions for *TensorShop*. Tell us about your experience using *TensorShop*, suggest some improvements, or reach out about *TensorShop* by emailing: .

js5384@columbia.edu josh@cs.columbia.edu

XIV. References

[Alm19] Alman, J. (2019). Linear algebraic techniques in algorithms and complexity (Doctoral dissertation, Massachusetts Institute of Technology).

[Blä13] Bläser, M. (2013). Fast matrix multiplication. Theory of Computing, 1-60.

[CU03] Cohn, H., & Umans, C. (2003, October). A group-theoretic approach to fast matrix multiplication. In 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings. (pp. 438-449). IEEE.

[CW87] Coppersmith, D., & Winograd, S. (1987, January). Matrix multiplication via arithmetic progressions. In Proceedings of the nineteenth annual ACM symposium on Theory of computing (pp. 1-6).

[CGLVW21] Conner, A., Gesmundo, F., Landsberg, J. M., Ventura, E., & Wang, Y. (2021). Towards a geometric approach to Strassen's asymptotic rank conjecture. Collectanea mathematica, 72(1), 63-86.

[Wil12] Williams, V. V. (2012, May). Multiplying matrices faster than Coppersmith-Winograd. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing (pp. 887-898).