Documentation for Differential Forms Summer Project

Moustafa Gharamti, Samuel Kirwin-Jones, Maciej Tomasz Jarema

0.1 Geometry of stack vectors

0.1.1 Translations needed to define stack vectors and arrowheads

Stack vectors are covariant vectors, defined by planar sheets (lines, when working in 2D) perpendicular to arrows of the contravariant vector. The density of these planes, is determined by magnitude of the vector field at each point in space. These stack vectors, being covariant, correspond to differential forms, while arrow vectors (being contravariant) correspond to vector fields.
In python, these stack sheets have to be defined from the magnitude and direction of the input, based on x and y components of the 1-from.

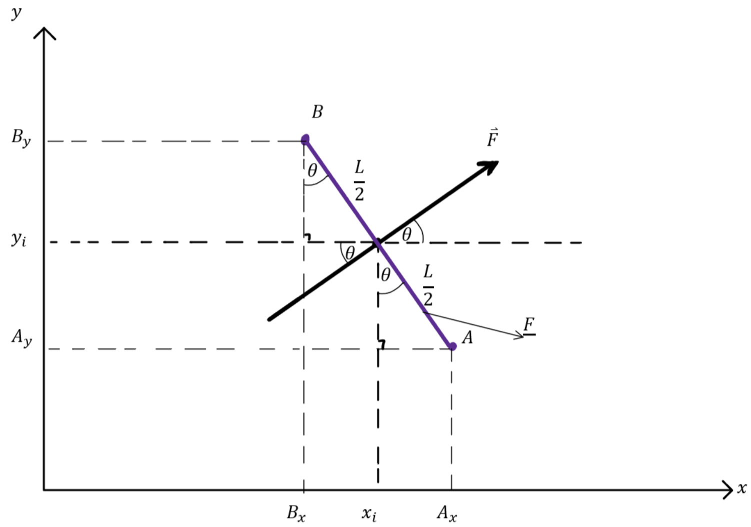

Figure 1: Sketch presenting the Geometrical arguments needed to define stack sheets, based on field magnitude and direction(indicated well by an arrow vector as shown)

For x_i and y_i marking the i^{th} considered position in the field, with its corresponding angle to the x-axis (ccw). Technically, θ, A. B and F as well as A and B components, depend on the position in space that is considered, therefore, should include the subscript, 'i'. This was not added for figure clarity The stack is shown in purple, with its endpoints, A and B, at a perpendicular distance to the arrow L (in the code, defined as a fraction of graph scale). From these, through simple geometry, one obtains the following equations:

$$A_x = x + \left(\frac{L}{2}\right)\sin(\theta)$$

$$A_y = y + \left(\frac{L}{2}\right)\cos(\theta)$$

$$B_x = x - \left(\frac{L}{2}\right)\sin(\theta)$$

$$B_y = y + \left(\frac{L}{2}\right)\cos(\theta)$$

describing positions of points A and B in terms of their Cartesian components.
These also function generally as operations that displace a point on the vector in the direction perpendicular to the arrow, by a corresponding length - here by $\frac{L}{2}$
To then displace the stack sheet in the direction parallel to arrow:

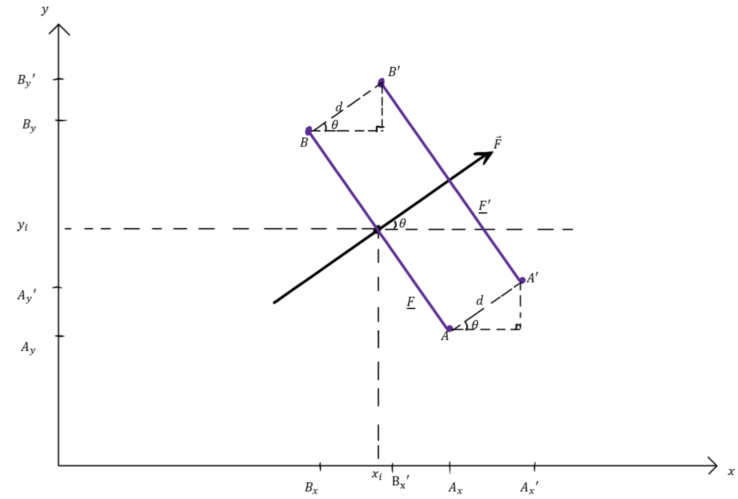

Figure 2: Sketch presenting the Geometrical arguments needed to displace stack sheets in the direction parallel to the vector field direction at that point, again - represented by an arrow)

Giving the following translation equations:

$$A'_x = A_x + d\cos(\theta)$$
$$A'_y = A_y + d\sin(\theta)$$
$$B'_x = B_x + d\cos(\theta)$$
$$B'_y = B_y + d\sin(\theta)$$

Which again function as general operations for such parallel displacements by any distance d, from the centre.

To define the arrowhead on that stack vector, both of these translation operations need to be used to obatin points to be connected as shown on the figure below

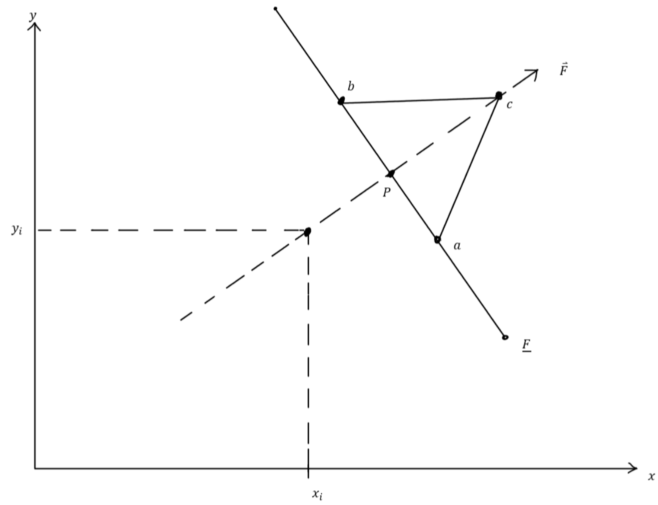

Figure 3: Sketch presenting the Geometrical arguments needed to define the arrowhead on each stack vector)

$$P_i = \left(x_i + \frac{L_s}{2}\cos(\theta),\ y_i + \frac{L_s}{2}\sin(\theta)\right)$$
$$a_i = \left(P_{x,i} + \frac{L_s}{w_{head}}\sin(\theta),\ P_{y,i} - \frac{L_s}{w_{head}}\cos(\theta)\right)$$
$$b_i = \left(P_{x,i} - \frac{L_s}{w_{head}}\sin(\theta),\ P_{y,i} + \frac{L_s}{w_{head}}\cos(\theta)\right)$$
$$c_i = \left(x_i + \frac{L_s}{h_{head}}\cos(\theta),\ y_i + \frac{L_s}{h_{head}}\sin(\theta)\right)$$

In these equations, variables are defined as shown on the above figure. L_s is the maximum length of the stack, parallel to the direction of the considered vector field, w_{head} is the width os the base of the arrowhead that rests on the last stack, and $hhead$ is the height os the arrowhead parallel to the vector field direction.

Note that each has also been assigned a subscript i which does not appear on the figure. This is only done as these coordinates depend on the point in space that is currently being considered. Also note, that for a single sheet in the stack, the initial displacement from $(x,\ y)$ to P must be omitted.

0.1.2 General from of stack sheet positions, for any number of sheets

In code, it was important to define the displacements of each stack sheet, form the middle position of the arrow vector, at each point in the field. Positioning of sheets depends on the magnitude of the vector field at that point in space. The larger the magnitude, the higher the sheet density. As the spread of the sheets is limited to a pre-defined maximum (on the plot, not physically), this increase in density corresponds to more stack sheets being present.

One way of plotting these, such that all sheets are equally-spaced is to consider odd and even number of stacks separately. We define them by displacing stack sheets along the arrow by certain lengths (as per the equations above), from the middle position (the considered position in the field).

For the odd number of stack sheets, the following pattern was noticed:
For 1 sheet: no displacing is needed $\Rightarrow \pm 0$
For 3 sheets: none, and ends of the stack $\Rightarrow \pm 0$ and $\pm\frac{1}{2} \longrightarrow$ technically $\pm\frac{1}{2}\cdot L_s$
For 5 sheets: none, ends, and points equally between $\Rightarrow \pm 0$, $\pm\frac{1}{2}$ and $\pm\frac{1}{4}$
For 7 sheets: $\Rightarrow \pm 0$, $\pm\frac{1}{2}$, $\pm\frac{2}{6}$ and $\pm\frac{1}{6} \longrightarrow$ At this point, note that $\frac{3}{6} = \frac{1}{2}$ (obvious but important)
For 9 sheets: $\Rightarrow \pm 0$, $\pm\frac{1}{2}$, $\pm\frac{3}{8}$, $\pm\frac{1}{4}$ and $\pm\frac{1}{8} \longrightarrow$ Again, $\frac{1}{4} = \frac{2}{8}$ and $\frac{1}{2} = \frac{4}{8}$
For 11 sheets: $\Rightarrow \pm 0$, $\pm\frac{1}{2}$, $\pm\frac{4}{10}$, $\pm\frac{3}{10}$, $\pm\frac{2}{10}$ and $\pm\frac{1}{10}$
Etc.

These display the following recursion:
The displacement along the arrow as a fraction of total stack length L_s, of the s$^{\text{th}}$ sheet, when drawing n stack sheets overall, for odd n is:

$$\pm\frac{s}{(n-1)}$$

Where we require the fractional displacement (from the middle position) to not exceed half of the total length therefore:

$$1 < s < \frac{1}{2}(n-1)$$

For the even number of stack sheets this pattern emerges:
2 sheets:$\Rightarrow \pm\frac{1}{2}$
4 sheets:$\Rightarrow \pm\frac{1}{2}$ and $\pm\frac{1}{6}$
6 sheets:$\Rightarrow \pm\frac{1}{2}$, $\pm\frac{3}{10}$ and $\pm\frac{1}{10}$ $\longrightarrow$ Note again, $\frac{1}{2} = \frac{5}{10}$, from here, continue with those not reduced fractions:
8 sheets: $\Rightarrow \pm\frac{1}{14}$, $\pm\frac{3}{14}$, $\pm\frac{5}{14}$ and $\pm\frac{7}{14}$
10 sheets: $\Rightarrow \pm\frac{1}{18}$, $\pm\frac{3}{18}$, $\pm\frac{5}{18}$, $\pm\frac{7}{18}$ and $\pm\frac{9}{18}$
Etc.

These display the following recursion:
The displacement along the arrow as a fraction of total stack length of the s$^{\text{th}}$ sheet, when drawing n stack sheets overall, for odd n is:

$$\pm\frac{2s+1}{2\,(n-1)}$$

Where we require the fractional displacement to not exceed half of the total length therefore:

$$1 < s < \frac{1}{2}\,(n-2)$$

Alternatively, each can be defined separately, manually, if a small amount of sheets is needed.

0.1.3 variables in stack plot code

This has been implemented in code. The parameters for this are as follows:

- L is the length of axis in each x and y, from origin
- pt_den is the number of points along each axis.
- a is a linear scaling of the field
- u is the x component $\longrightarrow$ Alternatively:
- v is the y component $\longrightarrow$ Fr is the radial component and $Ftheta$ is the angular component
- orientation is a sting that defined how arrows pivot
- scale is a linear scale on the quiver plot arrows
- delta is the extra length along the axis to show, past the defined grid to show full emerging arrows from border points
- fract is the fraction of graph length equal to stack sheet in direction perpendicular to arrow
- s_max is the maximum number of stacks to use
- sheet_L is the length of stack perp. to arrow
- s_L is the maximum length of stack sheet parallel to arrow
- w_head is the width of the arrowhead base as the denominator of the fraction of the stack sheet length perpendicular to the arrow
- h_head is the length of the arrowhead parallel to the arrow, as a denominator of the fraction of the total stack size parallel to the arrow

0.2 Initial GUI of the quiver and stack plot

0.2.1 Explaining user defined functions in the GUI code

parity: this function takes an input of an integer and returns True (1) is it is even and False (0) when it is odd. It is useful when defining stack sheets as displacements from the middle position, as the formulas are different for even and odd number of sheets per stack.
G: it takes three inputs $\Rightarrow$ s: the recursion of sheet displacing from the middle position (which pair is being completed), n: how many sheets are to plotted in total and c: which is the **bool** value from the parity function
stack_plot:takes the following inputs:

- xg and yg: the grid of points to be used when plotting the field
- ax: the axis to plot on
- u and v: the x and y components of the field to be plotted
- s_max: maximum number of sheets to plot, changes how dense the plot appears
- L: changes the size of axis (in both x and y equally, from origin)
- pt_den: defines the number of points on each axis that create the grid
- fract: defines the size of the stack sheet (as a square) as a fraction of total graph size
- arrows (optional, default=True): bool variable, defines if arrows should be plotted on top of the stacks (when True), or if only stacks are to be plotted (when False)
- orientation (optional, default='mid'): sets the pivoting point of the arrows about that grid point that they are defined at.
- scale (optional, default=1): linearly scales the arrows in the quiver plot
- w_head (optional, default=8): sets the denominator of the fraction that defines the width of the arrowhead at its base (on the stack), from the total size of the stack.
- h_head (optional, default=4): sets the denominator of the fraction that defines the height of the arrowhead parallel to the vector field magnitude at that point, from the total size of the stack.

on_key_press: function that tracks mouse key presses, needed for the 'Matplotlib' toolbar to function
format_eq: takes a single string, converts all variables in it that are common in vector field equations, and turns them into things that python can understand. Returns the corrected string
eq_to_comps: takes the two strings given by the user (equations for the field in the x and y directions) as well as the x and y grids, and the previous (or initial arrays for) u and v. Uses the above function (format_eq) to make the string 'python readable'. If one or more of the strings is zero, it defines a zero array for the component to be zero over all points along the grid and for shapes to match. Otherwise, it evaluates the given equation and returns the vector field components u and v
vect_type_response: Responds to changes in Radio-buttons that set the type of field to be plotted (arrow, stacks or both). Takes in a value from the Radio-buttons corresponding to the chosen field type. It clear the current plot. Checks which button has been selected and uses ax.quiver and previously user defined stack_plot to create the updated graph. It then updates it on the GUI by using canvas.draw() for the canvas being defined on the main window, in its own frame. Returns no variables.
PLOT_response:Responds to the 'PLOT' button being pressed. Updates the axis scaling, point density, maximum number of sheets per stack, linear scaling ('a') and the new field components. Takes in no

input. collects all needed variables by the '.get()' method of Tkinter objects. After running, it plots the new specified field, with the new parameters as a stack only plot and changes the status of the Radio-buttons to one again be - stacks only (therefore for tensor (same variable name as in VFA java code) to equal 0)