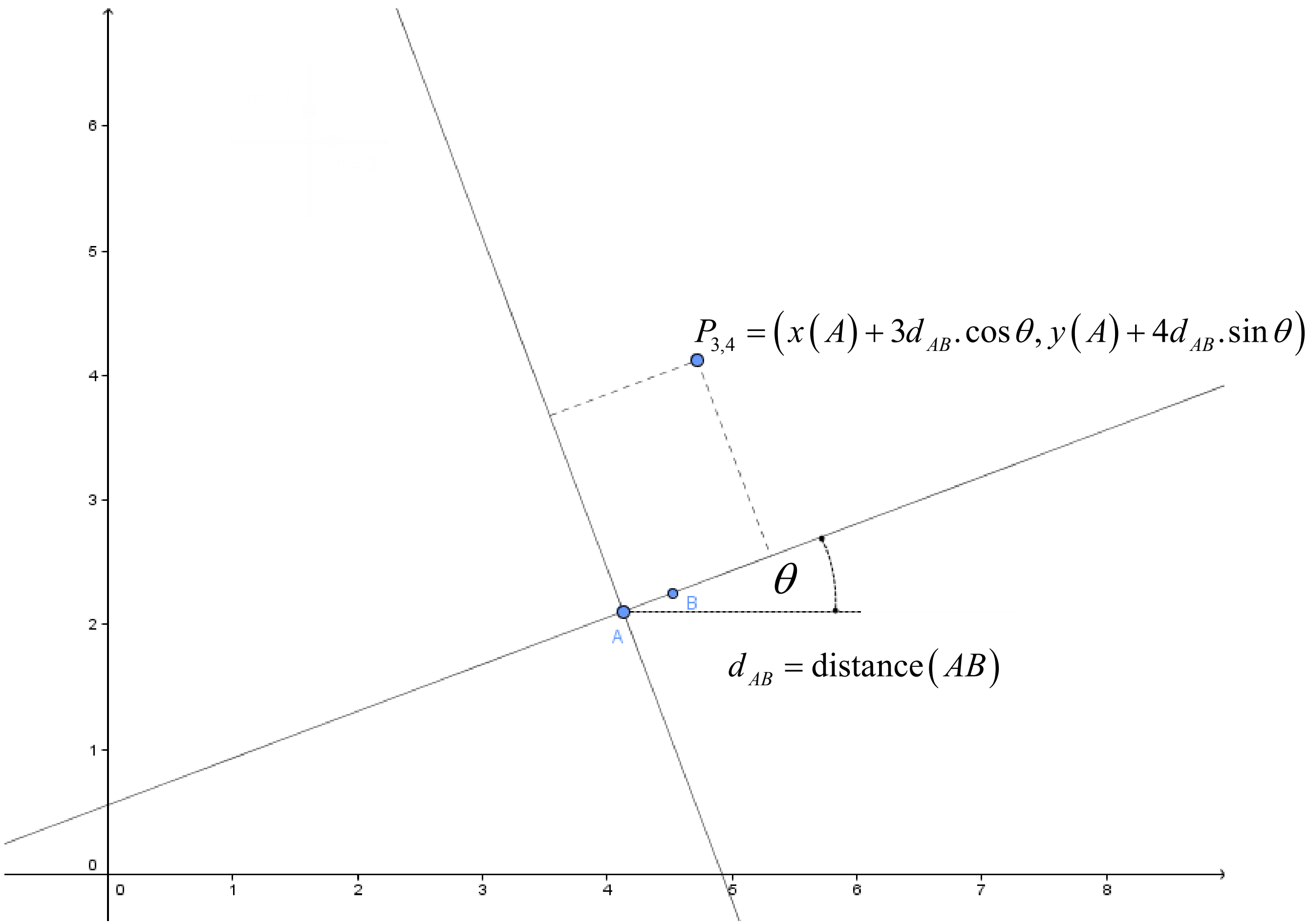
GeoGebra 4: Use of the Sequence Function to create a custom grid.

The following work illustrates the steps we took to produce a rectangular grid, centred at a movable point *A*, rotated at an angle θ to the horizontal and having a scale based upon the distance between points *A* and *B*, where *B* is a second movable point.

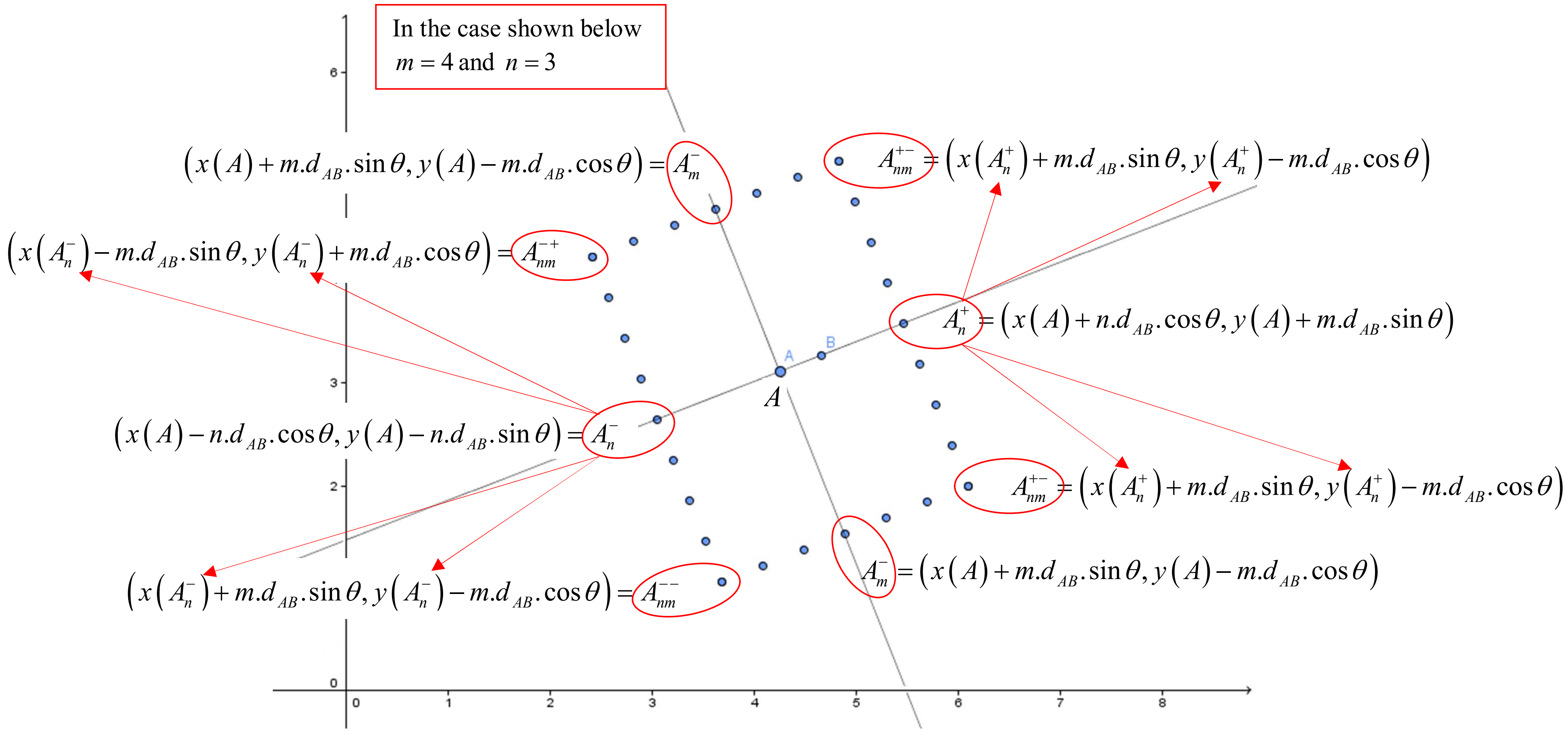
File "Grid1.ggb" contains the following in which a point P was set up to move integer multiples of the distance AB, parallel and perpendicular to the new axes through A, rotated at an arbitrary angle θ to the horizontal as determined by the slope of the line through A and B.

In this and subsequent files a slider named "n" will govern the multiples of AB along the axis through AB and a slider named "m" will govern the multiples of AB perpendicular to the axis through AB.



The formulae from files “Grid2.ggb” and “Grid3.ggb” are combined within the “Grid4.ggb” file and so we now look at part of this file in greater detail. Given that our object is to produce a rectangular grid, centred at *A*, rotated at an angle θ to the horizontal and having a scale based upon the distance between points *A* and *B*, then it would seem a simple matter to join opposite points with segments. However, we leave this until “Grid6.ggb” and later files…

Shown below is how the system of points has been constructed, based upon the original points A and B, the distance between them and their slope.



Formulae associated with this file:

* Represents text/formula having to be typed into the Input Bar.

Construction Protocol - Grid4.ggb

No.	Name	Definition	Value	Caption
1	Point A		A = (4.1, 2.1)	
2	Point B		B = (4.5, 2.3)	
3 *	Number d_{AB}	Distance[A, B]	d_{AB} = 0.4	
4 *	Number θ	atan(Slope[Line[A, B]])	θ = 0.4	
5	Line a	Line through A, B	a: -0.2x + 0.4y = 0.2	
6	Number m		m = 4	
7	Number n		n = 3	
8 *	List GridParallelABneg	Sequence[(x(A) - n d_{AB} cos(θ) - i d_{AB} sin(θ), y(A) - n d_{AB} sin(θ) + i d_{AB} cos(θ)), i, -m, m]	GridParallelABneg = {(3....	
9 *	List GridParallelABpos	Sequence[(x(A) + n d_{AB} cos(θ) - i d_{AB} sin(θ), y(A) + n d_{AB} sin(θ) + i d_{AB} cos(θ)), i, -m, m]	GridParallelABpos = {(5....	
10 *	List GridPerpABpos	Sequence[(x(A) - m d_{AB} sin(θ) + j d_{AB} cos(θ), y(A) + m d_{AB} cos(θ) + j d_{AB} sin(θ)), j, -n, n]	GridPerpABpos = {(2.3, 3...	
11 *	List GridPerpABneg	Sequence[(x(A) + m d_{AB} sin(θ) + j d_{AB} cos(θ), y(A) - m d_{AB} cos(θ) + j d_{AB} sin(θ)), j, -n, n]	GridPerpABneg = {(3.6, 0...	
12	Line b	Line through A perpendicular to a	b: -0.4x - 0.2y = -2	

12 / 12

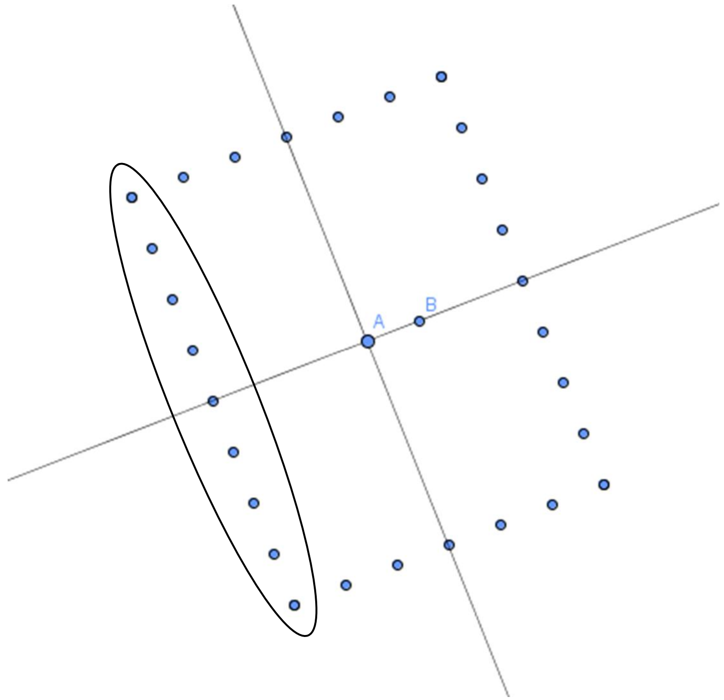
As an example, Step 8 above in the Construction Protocol Box would be entered thus into the Input Bar:

```
GridParallelABneg = Sequence[(x(A)-n*d_{AB}* cos(θ)-i* d_{AB}* sin(θ),y(A)-n*d_{AB}* sin(θ)+i* d_{AB}* cos(θ)),i,-m,m]
```

For programmers (especially so VBA), this single GeoGebra line could be visualised as the following *pseudocode* control loop:

```
------------------------------------
    m = 4
    n = 3
    xA = 4.2  'This just happens to be the current x-coordinate of the point A
    yA = 3.1  'This just happens to be the current y-coordinate of the point A
    dAB = 0.4   'This just happens to be the current distance between the points A and B
    Theta = 0.4   'This just happens to be the current angle in radians between the segment AB and the horizontal

        For i = -m to m   'A step size of 1 is implied if not typed in
            xCoord = xA – n*dAB*Cos(Theta) – i*dAB*Sin(Theta)
            yCoord = yA – n*dAB*Sin(Theta) + i*dAB*Cos(Theta)
        Next i
------------------------------------
```



The Points that are generated by this loop are shown highlighted here:
i = -m (or -4 here) generates the point at the bottom left corner.

Note that to generate the set of points parallel to these on the other side of the rectangle the "Grid4.ggb" file used the following Input Bar entry:

GridParallelABpos = Sequence[(x(A)+n*d_{AB}* cos(θ)-i* d_{AB}* sin(θ),y(A)+n*d_{AB}* sin(θ)+i* d_{AB}* cos(θ)),i,-m,m]

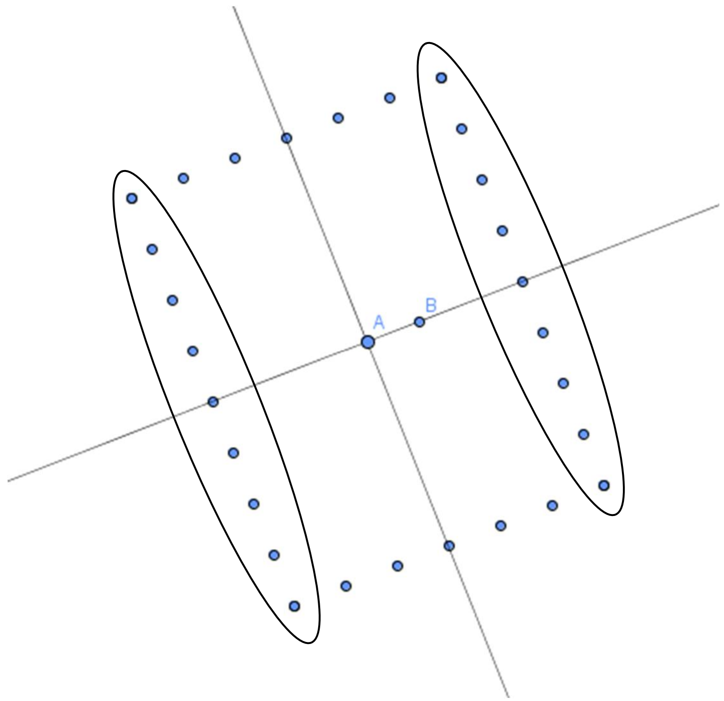
As you may see there is very little difference between these two formulae, therefore, in the file "Grid4AlternativeMethod.ggb", we combine these two loops into one in the following **single**-line formula:

GridParallelABpos =

Sequence[Sequence[(x(A) + j*d_{AB}*cos(θ) – i*d_{AB}*sin(θ), y(A) + j*d_{AB}*sin(θ) + i*d_{AB}*cos(θ)), i, -m, m], j, -n, n, 2n]

Just considering the loop structure in the above *pseudocode* we see that the extra loop below now generates both sets of points as highlighted in the picture here:

```
-------------------------------------
For j = -n to n, Step 2n
    For i = -m to m   'A step size of 1 is implied if not typed in
        xCoord = xA + j*dAB*Cos(Theta) – i*dAB*Sin(Theta)
        yCoord = yA + j*dAB*Sin(Theta) + i*dAB*Cos(Theta)
    Next i
Next j
-------------------------------------
```



Notice now that once this 'double-loop' structure has been employed then it is a simple matter to modify the loop arguments to draw every boundary grid point and every internal grid point of the rectangle, relative to the distance AB, as is seen in the file "Grid5.ggb".

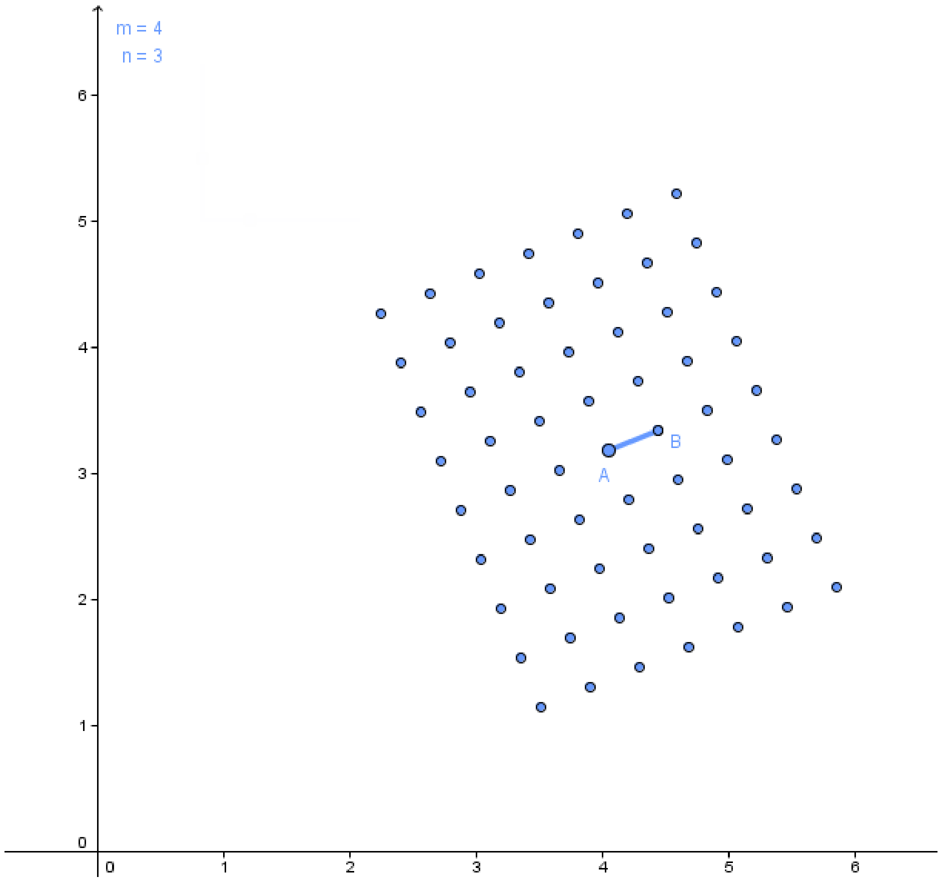
Here is the modification:

GridAllAB = Sequence[Sequence[(x(A) + j*d_{AB}*cos(θ) – i*d_{AB}*sin(θ), y(A) + j*d_{AB}*sin(θ) + i*d_{AB}*cos(θ)), i, -m, m], j, -n, n]

This single line equates to the following *pseudocode:*

```
------------------------------------
    For j = -n to n   'A step size of 1 is implied if not typed in
        For i = -m to m   'A step size of 1 is implied if not typed in
            xCoord = xA + j*dAB*Cos(Theta) – i*dAB*Sin(Theta)
            yCoord = yA + j*dAB*Sin(Theta) + i*dAB*Cos(Theta)
        Next i
    Next j
------------------------------------
```

and it generates the grid (or lattice) points as shown here:



In file "Grid6.ggb" we return to the task at hand, drawing a grid, and begin (w.l.o.g) by constructing segments parallel to AB

```
GridAllParallelABsegments= Sequence[Segment[(x(A) – n*d_{AB}*cos(θ) – i*d_{AB}*sin(θ), y(A) – n*d_{AB}*sin(θ) + i*d_{AB}*cos(θ)), (x(A) + n*d_{AB}*cos(θ) – i*d_{AB}*sin(θ), y(A) + n*d_{AB}*sin(θ) + i*d_{AB}*cos(θ))], i, -m, m]
```

Giving this single-line formula a little more structure we can see how the Sequence and Segment functions combine:

We have the function syntax:

```
Segment[ <Point>,<Point> ]
```

and

```
Sequence[ <Expression>, <Variable>, <Start Value>, <End Value> ]
```

Therefore, on nesting:

```
Sequence[Segment[ <Point>,<Point> ],<Variable>, <Start Value>, <End Value> ]
```

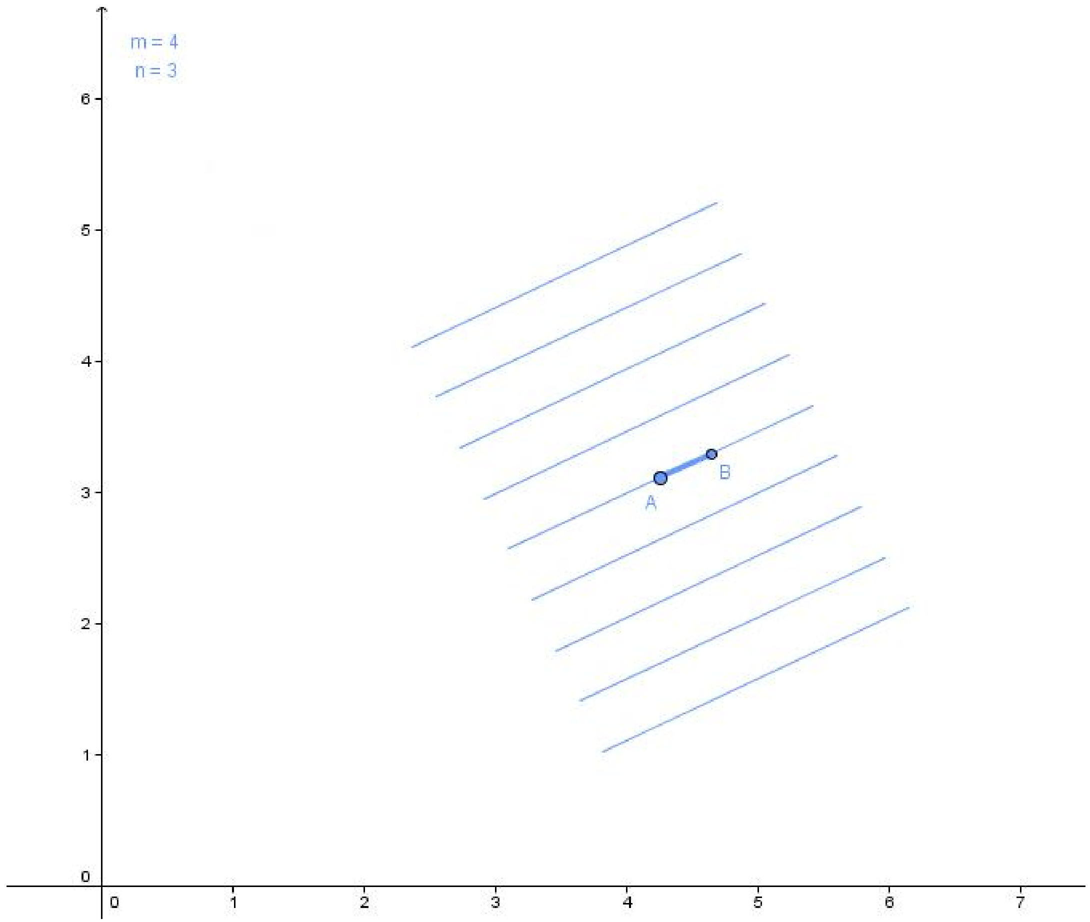
```
i, -m, m]
```

```
(x(A) – n*d_{AB}*cos(θ) – i*d_{AB}*sin(θ), y(A) – n*d_{AB}*sin(θ) + i*d_{AB}*cos(θ))
```

```
(x(A) + n*d_{AB}*cos(θ) – i*d_{AB}*sin(θ), y(A) + n*d_{AB}*sin(θ) + i*d_{AB}*cos(θ))
```

The construction of this set of points, the left hand side of the rectangle, was discussed in depth on page 5 above.

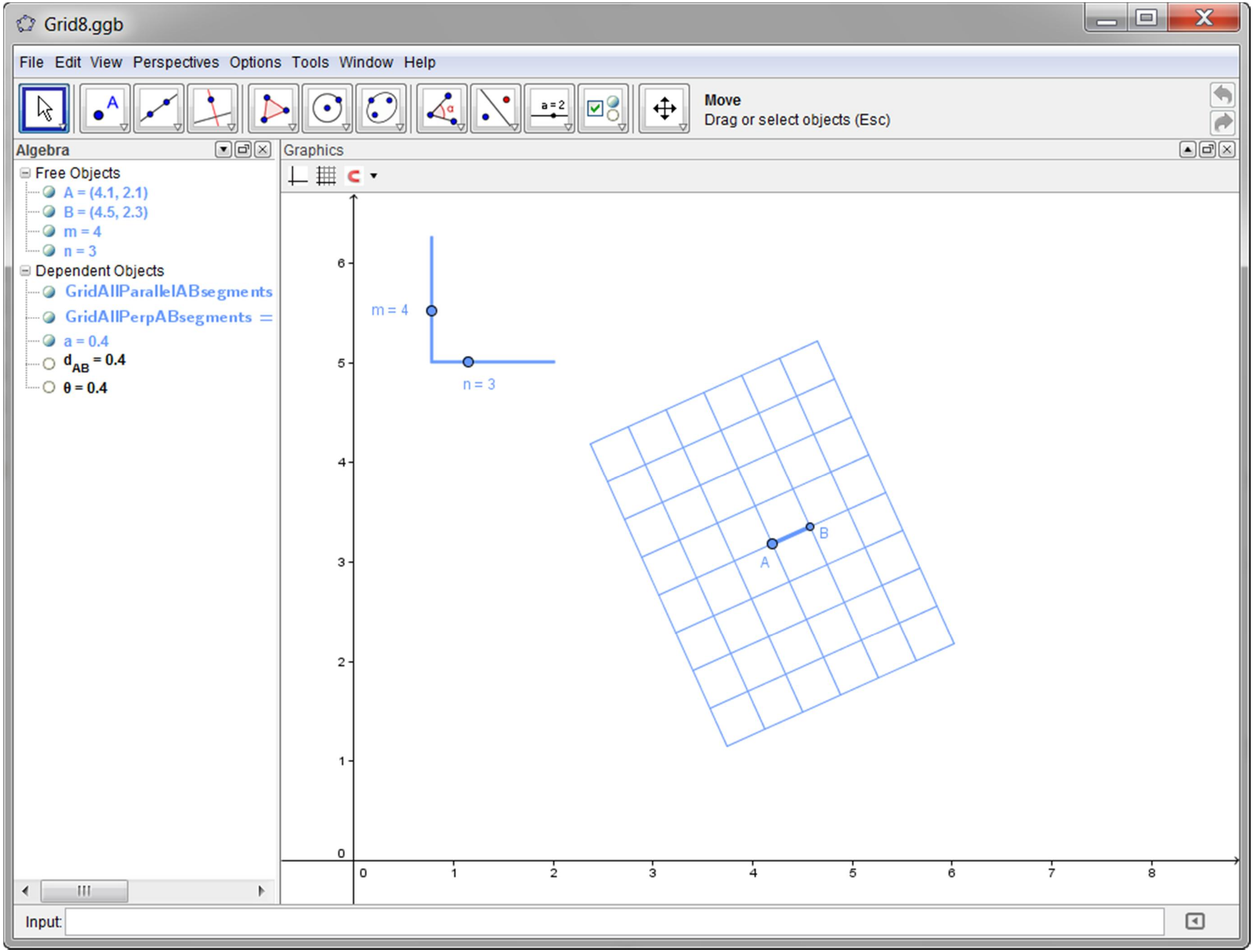
The above formula gave rise to the following picture:



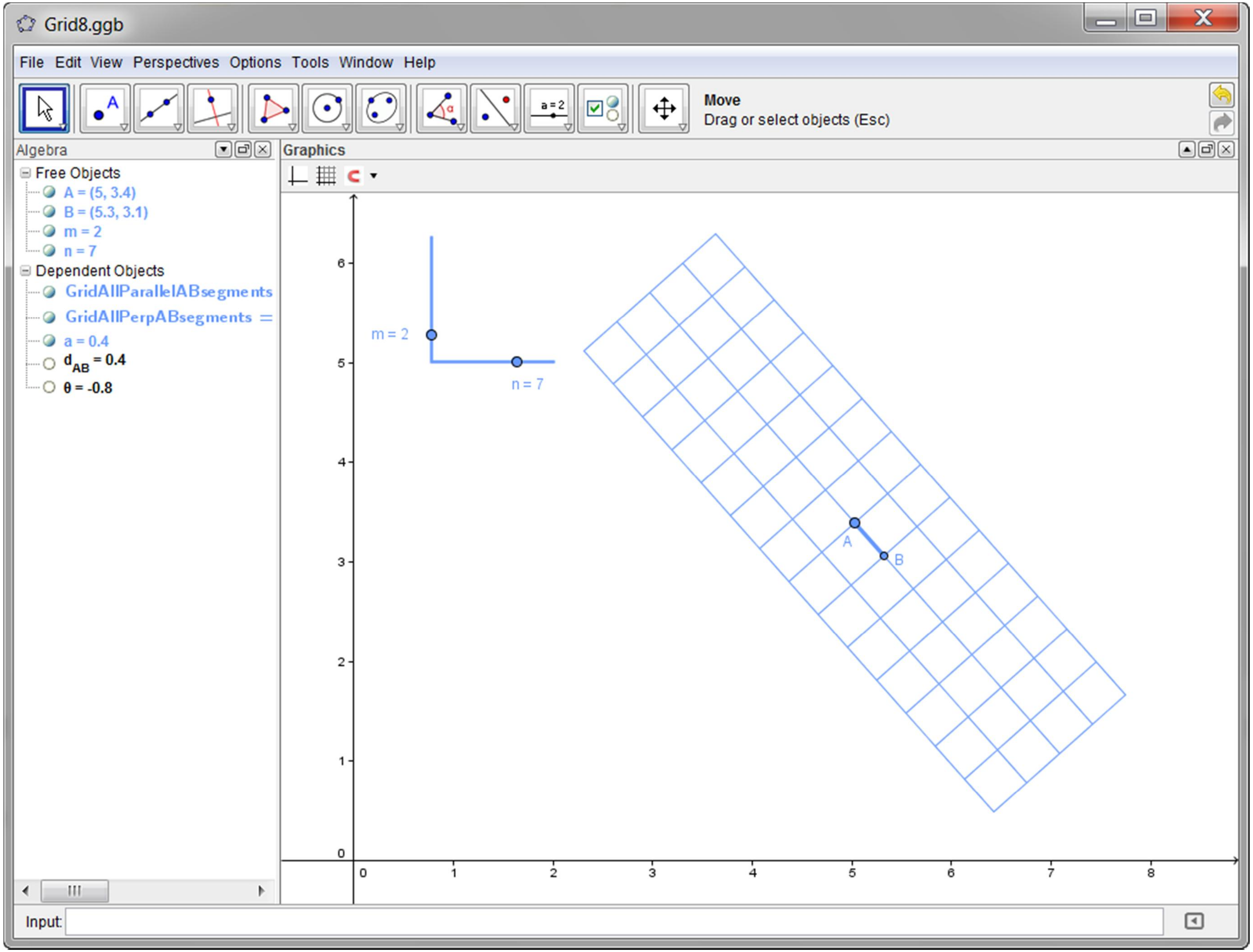
In a similar manner, from the file "Grid7", we use the following, single-line formula to construct segments perpendicular to AB

GridAllPerpABsegments = Sequence[Segment[(x(A) + j*d_{AB}*cos(θ) – m*d_{AB}*sin(θ), y(A) + j*d_{AB}*sin(θ) + m*d_{AB}*cos(θ)), (x(A) + j*d_{AB}*cos(θ) + m*d_{AB}*sin(θ), y(A) + j*d_{AB}*sin(θ) – m*d_{AB}*cos(θ))], j, -n, n]

Combining these two formulae into a single file, "Grid8.ggb", give the following output:



Here is another screenshot having altered parameter values and positions of A and B



In file "Grid9.ggb" we now seek to fill in this completed grid with subdivisions based upon further slider values

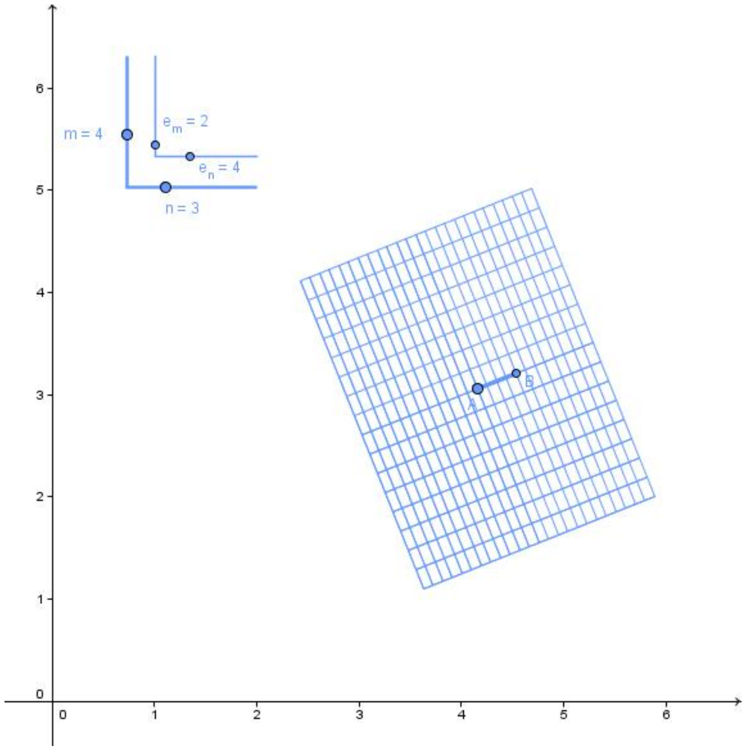
After setting up two more sliders, called e_m and e_n it is actually a simple matter to add a sub-grid.

There are two possible forms of syntax for the Sequence function. The grid in the file "Grid8.ggb" used the form:

Sequence[<Expression>, <Variable>, <Start Value>, <End Value>]

To create a sub-grid we use the alternative syntax having the Increment option:

Sequence[<Expression>, <Variable>, <Start Value>, <End Value>, <Increment>]



We then set increments to be 1 / e_m and 1 / e_n (that is $\frac{1}{e_m}$ and $\frac{1}{e_n}$), respectively

in the GridAllParallelABsegments and GridAllPerpABsegments functions in the file "Grid9.ggb".

One final modification was used in the file "Grid10.ggb" to overcome the whole grid vanishing when point B is either vertically above or below A. This effect is due to the discontinuity in the Slope function when the angle θ takes values of either $\pi/2$ or $-\pi/2$.

Original definition:

$$\theta = \text{atan}(\text{Slope}[\text{Line}[A, B]])$$

Modified definition: (Using three nested IF functions)

$$\theta = \text{If}[\underbrace{x(A) \neq x(B)}_{\text{Condition that B is not verticaly above A}}, \text{atan}(\text{Slope}[\text{Line}[A, B]]), \text{If}[\underbrace{x(A) \stackrel{?}{=} x(B) \wedge (y(B) > y(A))}_{\text{Condition that B is verticaly above A}}, \pi/2, \text{If}[\underbrace{x(A) \stackrel{?}{=} x(B) \wedge (y(B) < y(A))}_{\text{Condition that A is verticaly above B}}, -\pi/2]]]$$

In file "Grid10 - Application.ggb" we see an application of the grid to a planar reflection problem. We can imagine the grid as a sheet of tracing paper onto which a grid has already been printed. In this manner we can use the grid, suitably adjusted, to better judge the perpendicular distance of the shape given to the mirror line as shown below: