

Field lines

This is a description of the code behind the calculations of field lines presented in class. This is not required material for the course, but it may be of interest to those with a bent for computer modeling. The details of the code are definitely specific to the *Mathematica* language, so some of what is presented below may be opaque. The sequence of steps, however, is typical of what is needed to do calculations of this type. For those with the inclination to follow up, developing a modeling code usually forces you to come to a better understanding of your material.

■ Illustration of field lines

At each point in space there is an electric field $\mathbf{E}$. This is a vector quantity with direction and magnitude. One can make a map of the electric field by laying out an array of vectors. An alternative way to visualize the field is to specify field lines. The prescription for finding the field line which goes through a point p is,

1. Let the position along the field line be $\mathbf{x}$
2. Start at position $\mathbf{x} = p$
3. Find $\mathbf{E}(\mathbf{x})$
4. Take a small step of length $d\mathbf{t}$ in the direction of $\mathbf{E}$
5. Assign the new position as $\mathbf{x}_{\text{new}} \rightarrow \mathbf{x}_{\text{old}} + \delta\mathbf{x} = \mathbf{x}_{\text{old}} + \mathbf{E} d\mathbf{t}$
6. Repeat 3-5 until done

This specification is equivalent to saying the field line is determined by the differential equation $\frac{d\mathbf{x}}{dt} = \mathbf{E}$, where $\mathbf{x}$ and $\mathbf{E}$ are both vectors. There are many numerical methods for solving such equations, some of which are built into *Mathematica*. Thus, the steps for producing a field line illustration are

1. Select the charges q_i and their positions, $\mathbf{x}_i$.
2. Select the points $\mathbf{p}_j$ through which the field lines will pass
3. Provide a routine for calculating $\mathbf{E}(q_i, \mathbf{x}_i, \mathbf{p}_j)$
4. Give the charges and points to the routine which does the solution of $\frac{d\mathbf{x}}{dt} = \mathbf{E}$ producing one field line for each point $\mathbf{p}$.
5. Make an illustration of the field lines.

The code below gives an example of how this may be done in practice.

■ code

Almost any program needs to be initialized and provide access to a library of methods for performing useful tasks. Similarly, it is usually necessary to write a few utility methods which perform essential tasks, but are not the focus of the problem at hand. I'll hide both these items.

■ init

■ utilities

■ electric field

We need to calculate the electric field from a collection of charges. This is typically done by finding the electric potential and taking its gradient $\mathbf{E} = \nabla \Phi$, but at this point in the course we haven't yet introduced potentials, so the code here calculates $\mathbf{E}$ the old fashioned way, by finding $\mathbf{E} = \sum_i \mathbf{E}_i$ as a sum over the contribution from each charge in the collection.

Here is the method for calculating the electric field at $\mathbf{x}_2$ due to charge q at $\mathbf{x}_1$. In this code ($\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{dx}$) are all vectors. The norm function produces a vector of unit length in the direction $\mathbf{dx}$. The returned field is seen to be $\mathbf{E} = \frac{q}{r^2} \hat{\mathbf{r}}$.

```
In[7]:= el2[{q_, x1_}, x2_] := Block[{},
    dx = x2 - x1;
    q / (dx.dx) norm[dx]]
```

The next method produces a normalized electric field direction at point $\mathbf{x}_2$ due to a collection of charges qs . The syntax may be obscure, but you can see that the electric field method `el2` is accessed, and the result from many charges is added by the Plus function. The result is normalized, for a couple of reasons. First, many differential equation solvers work better if the value of the derivative is not varying by large values. Second, if the direction vector is normalized, then the length of the field line is easily calculated during the integration. A disadvantage is that if one was interested in the line integral $\int \mathbf{E} \cdot d\mathbf{l}$, then one would want the direction vector to have the magnitude of the electric field.

```
In[8]:= ehat[qs_, x2_] := norm[Plus @@ (el2[#, x2] & /@ qs)]
```

The next method solves the differential equation $\frac{d\mathbf{x}}{dt} = \hat{\mathbf{E}}$ for a field line which passes through the point $\mathbf{xx0}$. It stops integrating if the field line hits a charge or escapes the test volume. The method is set to extend the line in both directions from $\mathbf{xx0}$. The field line is returned as three functions which specify the three coordinates as a function of position along the field line. The function "check" is true if the end condition has been achieved.

```
In[9]:= eline[qs_, xx0_] := Block[{},
    {x0, y0, z0} = xx0;
    sol = NDSolve[{x'[t] == ehat[qs, {x[t], y[t], z[t]}][[1]],
        y'[t] == ehat[qs, {x[t], y[t], z[t]}][[2]],
        z'[t] == ehat[qs, {x[t], y[t], z[t]}][[3]], x[0] == x0,
        y[0] == y0, z[0] == z0}, {x, y, z}, {t, -100, 100}, Method ->
        {"EventLocator", "Event" :> check[qs, {x[t], y[t], z[t]}]}];

    {x$, y$, z$} /. sol[[1]]]
```

Finally, the method to find the field lines for a number of points.

```
In[10]:= elines[qs_, pnts_] := ({x[#], y[#], z[#]} = eline[qs, #]) & /@ pnts;
```

Next, there are some more utilities to make illustrations, which I will also hide.

■ Utilities to make select points make plots etc

■ Cases

■ dipole

Define list of charges

```
In[18]:= q1 = {1, {0, 2, 0}};
q2 = {-1, {0, -2, 0}};
qs = {q1, q2};
```

Get initial points to start field lines.

```
In[21]:= pnts = Flatten[getpnts[qs, .1], 1];
```

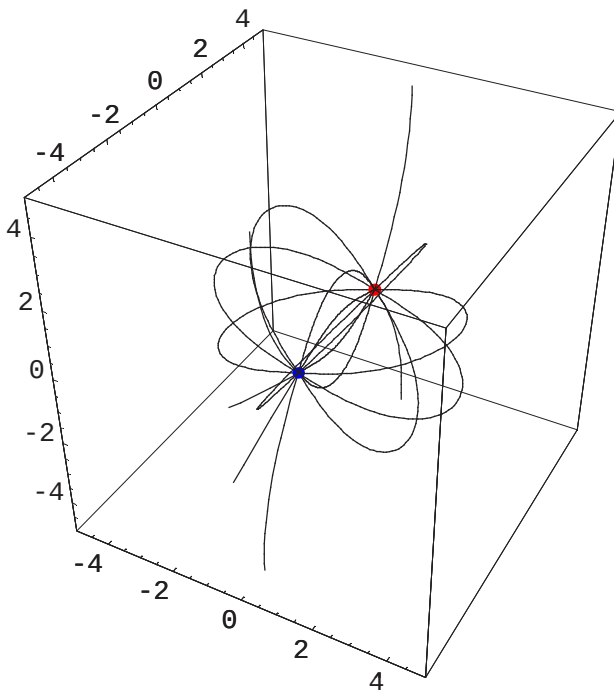
Do calculations including turning field lines into an illustration.

```
In[22]:= Timing[p[1] = makeelineplot[qs, pnts];]
```

```
Out[22]= {0.771 Second, Null}
```

And show the result

```
In[25]:= Show[p[1], show];
```



More examples follow, but I'll just show the results using *Mathematica's* web interface.