About the load vector for edge elements.

The time-harmonic model for $\mathbf{E}(x,y)$ (assuming $\mathbf{H} = H\mathbf{e}_z$ and PEC / PMC boundary so boundary terms vanish) is (book formula 6.46)

$$\int_S \{\mu^{-1}(\nabla\times\mathbf{N}_i)\cdot(\nabla\times\mathbf{E}_i) - \omega^2\varepsilon\mathbf{N}_i\cdot\mathbf{E}\}dS = -j\omega\int_S \mathbf{N}_i\cdot\mathbf{J}dS$$

This is the case treated by the codes from the book. If instead we assume $\mathbf{E} = E\mathbf{e}_z$, we obtain

$$\int_S \{\varepsilon^{-1}(\nabla\times\mathbf{N}_i)\cdot(\nabla\times\mathbf{H}) - \omega^2\mu\mathbf{N}_i\cdot\mathbf{H}\}dS = -\int_S \varepsilon^{-1}(\nabla\times\mathbf{N}_i)\cdot J\mathbf{e}_z dS$$

We shall discuss the load vector calculation. The forms above are necessary for inhomogeneous wave guides with regions with different ε. For the case of constant ε (μ never deviates from μ_0 except for ferromagnetic materials) we have

$$\int_S \{(\nabla\times\mathbf{N}_i)\cdot(\nabla\times\mathbf{E}_i) - k^2\mathbf{N}_i\cdot\mathbf{E}\}dS = -j\mu\omega\int_S \mathbf{N}_i\cdot\mathbf{J}dS \qquad (1)$$

and

$$\int_S \{(\nabla\times\mathbf{N}_i)\cdot(\nabla\times\mathbf{H}) - k^2\mathbf{N}_i\cdot\mathbf{H}\}dS = -\int_S (\nabla\times\mathbf{N}_i)\cdot J\mathbf{e}_z dS \qquad (2)$$

For (1), because div $\mathbf{J} = 0$, we represent the current density as

$$\mathbf{J} = \sum_i J_i\mathbf{N}_i$$

and must compute the load vector by assembling contributions from all triangles T_k,

$$\int_{T_k} \mathbf{N}_i\cdot\mathbf{N}_j dS$$

These element matrices have all been computed in the mass matrix *M*.

For (2) it is more natural to define the scalar current density in the nodal basis functions as

$$J = \sum_i J_i\varphi_i$$

and assemble

$$\int_{T_k} \nabla\times\mathbf{N}_i\cdot\varphi_j\mathbf{e}_z dS = \nabla\times\mathbf{N}_i\cdot\mathbf{e}_z\int_{T_k}\varphi_j dS = \frac{1}{3}A_k\nabla\varphi_m\times\nabla\varphi_n\cdot\mathbf{e}_z$$

where the ingredients have been calculated in the stiffness matrices `sloc`. For (2), the delta-function current can be provided by a thin wire in the *z*-direction, for (1) by a conducting sheet whose intersection with the (x,y)-plane forms a closed curve, following the element edges.

You r job is to implement the delta-current for case (2) - well, also a continuous $J(x,y)\mathbf{e}_z$ since that is what gave the problems, see below - and run a sweep of *k*-values like for part 1 of the lab.

Addendum Oct 29

The problems in the load vector construction is a problem of signs as we all thought. But it was hard for you to find the problem because the **`plot_field`** function had a bug, so visualization of the curl **J** field and the actual load vector provided confusing information. However, the plot bug was easy to find because the tangential component was NOT continuous across element boundaries. Thank you Daniele for showing me your zoomed field plot! Now the plot routine becomes a prime suspect because we know that any combination of the $\mathbf{N}_k$-basis functions must have tangential field continuous. You can see the bug comment in the code snippet provided below.

```
% E = sum sol(j) Nj, linearly interpolated, barycentric coord.,
% to finer mesh
Ex = phi_1*( grad_phi_2x.*sol1 + grad_phi_3x.*sol2) + ...
     phi_2*(-grad_phi_1x.*sol1 + grad_phi_3x.*sol3) + ...
     phi_3*(-grad_phi_1x.*sol2 - grad_phi_2x.*sol3);
%=== bug fixed: two last lines should have phi_2 and phi_3, not
phi_1 !!
Ey = phi_1*( grad_phi_2y.*sol1 + grad_phi_3y.*sol2) + ...
     phi_2*(-grad_phi_1y.*sol1 + grad_phi_3y.*sol3) + ...
     phi_3*(-grad_phi_1y.*sol2 - grad_phi_2y.*sol3);
```

We see that if the current is just a delta-function (e.g. a current =1 in only one triangle) the solution, for small k^2, should look like magnetostatics with field lines circling round the wire. The solutions for this case look OK. Next step is to make the current constant inside a circular disk, zero outside: a thick wire. Now curl **J** becomes a delta-function at the jump, so we expect visualization of curl **J** to show essentially zero inside and outside and large values at the circle perimeter. But – the pictures are very different, a chaotic pattern inside and zero outside. Neighbor triangles have **E**-fields of different signs so we suspect the signs, because we have repeatedly computed curl $\mathbf{N}_k$ and found the same answer over and again. So … think again:

1. The ordering of nodes to make the direction of triangle edges consistent between neighbors gives the signs +1,-1,+1 for basis functions $\mathbf{N}_1$, $\mathbf{N}_2$, $\mathbf{N}_3$. We see these signs in the **`edgeFEM2D`** local stiffness matrix **`s00`** which is just **`[1;-1;1]*[1,-1,1]`** in matlab notation, so that must be right.

2. The cross product of the phi-gradients equals + or – 1/area depending on the orientation of edges. This is a different issue than 1, but it does not show up in the calculation of stiffness matrix because that is a product of two curl **J** on the same triangle, so *this* sign is immaterial in **`edgeFEM2D`**!

There is in **`edgeFEM2D`** a **`detJ`**-array which is edge1 x edge2 for each triangle, so the final sign pattern is like

```
     sign(detJ(k))*[1,-1,1]’
```

for $\mathbf{N}_1$, $\mathbf{N}_2$, $\mathbf{N}_3$ in triangle k.

That was missing. So let **`edge2FEM2D`** return also **`detJ`** and use it in the load vector calculation. Here is a picture which shows, left to right, the load vector visualized as if its component k is the coefficient of $\mathbf{N}_k$ (not true, but almost. Why??), and the load vector more correctly visualized. Below the solution is blown up to show the field lines.

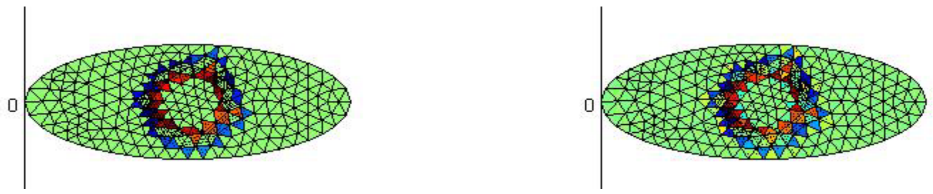

Fig. 1: Visualization of load vector, left, naïve, right, as Galerkin projection (see below)

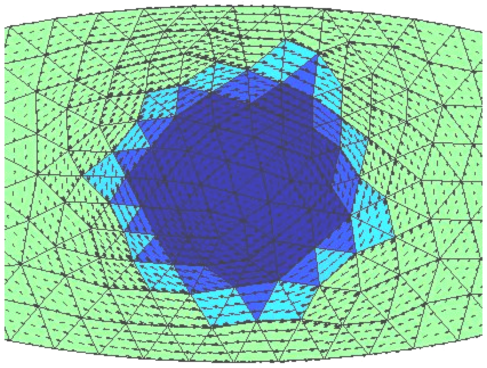

Fig. 2 **H-**field as arrows and E-field as color for *J* = 1 inside a circle, = 0 outside.

About visualization of curl (*J*(*x*,*y*)e$_z$)

This is a divergence free vector, so we expand it in the edge basis functions and compute the coefficients by Galerkin:

$$\nabla \times J\mathbf{e}_z = \sum j_k \mathbf{N}_k \Rightarrow \underbrace{\int_\Omega \mathbf{N}_m \cdot (\nabla \times J\mathbf{e}_z) d\Omega}_{=-\int_\Omega \nabla\times\mathbf{N}_m J d\Omega} = \sum j_k \underbrace{\int_\Omega \mathbf{N}_m \cdot \mathbf{N}_k d\Omega}_{M_{mk}} \cdot, m = 1,2,\ldots, N_{edge}$$

$$\mathbf{Mj} = \mathbf{f}$$

We must solve the linear system with mass matrix as coefficients to get something that can rigorously be visualized by **`plot_field`**. No problem, we have **M** from the assembly. But … **f** is also a vector associated with the edges, so can be sent to **`plot_field`**, as done in fig. left above. Why so similar to the "correct" picture?