Supplementary Material to the article "Topological memory with multiply-connected planar magnetic nanoelements"

We start with the following trial function for the magnetization distribution inside the circular cylinder, which is obtained from [PRL 105 (2010), 107201] by a series of variable substitutions, designed to limit the vortex core size variation (at a fixed parameter s) as its position is changed via the parameter $t \in [0, 1]$. At $t = 0$ the vortex is centered in the cylinder, at $t = 1$ it is split into two halves, which are at the opposite sides of the cylinder's face. The complex phase of t can control the precise location on the cylinder's boundary, where the vortex splitting takes place. For real t, which will be assumed in the following the vortex is split at $z = I$ and its two halves are at $z = \pm 1$ for $t = 1$.

In[ ]:=
```
trialf = FullSimplify[
    I z c + A - Conjugate[A] z^2 /. A → c a/2 /. c → s/(1 + a) /. a → t/(1 - t), s > 0];
Print[Limit[trialf , t → {0, 1/2, 1}]];
trialf = Function[{z, s, t}, Evaluate[trialf]];
trialf
```

$$\left\{i\, s\, z,\ \frac{1}{2}\, s\left(\frac{1}{2} + i\, z - \frac{z^2}{2}\right),\ \frac{1}{2}\, s\left(1 - z^2\right)\right\}$$

Out[ ]=

$$\text{Function}\left[\{z, s, t\},\ \frac{1}{2}\, s\left(t - 2\, i\, (-1 + t)\, z - z^2\, \text{Conjugate}[t]\right)\right]$$

Let us now introduce the piecewise expression for the complex function w and a simple Mathematica machinery to build expressions on top of it (function piecewiseExpr). This will allow us to define the various components, needed for computing the magnetic energies: the sum of squared gradients of the components of the magnetization vector (function sumgradsq), divergence of the magnetization vector m (function div), the m_X and m_y magnetization components, represented as a complex number (function mxmy) and the m_z component of the magnetization (function mz).

In[•]:=
```
piecewiseExpr[f_, d_, expr_] := Block[{fc, fa, wsi, wsic, wm, wmc, wso, wsoc},
    fc = Function[{z, zc}, Evaluate[FullSimplify[ComplexExpand[Conjugate[f[z, zc]], {_},
          TargetFunctions → Conjugate] /. {Conjugate[z] → zc, Conjugate[zc] → z}]]];
    fa = Function[{z, zc}, Re[f[z, zc] * fc[z, zc]]];
    wsi = f;
    wsic = fc;
    wm = Function[{z, zc}, f[z, zc] / Sqrt[f[z, zc] × fc[z, zc]]];
    wmc = Function[{z, zc}, fc[z, zc] / Sqrt[f[z, zc] × fc[z, zc]]];
    wso = Function[{z, zc}, f[z, zc] / d];
    wsoc = Function[{z, zc}, fc[z, zc] / d];
    Piecewise[{{expr[wsi, wsic], fa[z, zc] < 1},
      {expr[wm, wmc], 1 ≤ fa[z, zc] ≤ d}, {expr[wso, wsoc], fa[z, zc] > d}}]];
wfun[f_, d_] := piecewiseExpr[f, d, Function[{w, wc}, w[z, zc]]];
sumgradsq[f_, d_] := piecewiseExpr[f, d, Function[{w, wc}, 8 / (1 + w[z, zc] × wc[z, zc]) ^ 2
     (D[w[z, zc], z] × D[wc[z, zc], zc] + D[w[z, zc], zc] × D[wc[z, zc], z])]];
div[f_, d_] :=
  piecewiseExpr[f, d, Function[{w, wc}, 2 / (1 + w[z, zc] × wc[z, zc]) ^ 2 (D[w[z, zc], z] +
       D[wc[z, zc], zc] - w[z, zc] ^ 2 × D[wc[z, zc], z] - wc[z, zc] ^ 2 × D[w[z, zc], zc])]];
mxmy[f_, d_] := piecewiseExpr[f, d, Function[{w, wc}, 2 w[z, zc] / (1 + w[z, zc] × wc[z, zc])]];
mz[f_, d_] :=
  piecewiseExpr[f, d, Function[{w, wc}, (1 - w[z, zc] × wc[z, zc]) / (1 + w[z, zc] × wc[z, zc])]];
```

Here is the code to export the vector field for plotting .

```
In[•]:= Clear[mxmyForExport, mzForExport, exportField];
mxmyForExport[t_?NumericQ, s_?NumericQ, n_?IntegerQ] := With[
   {mxmy = mxmy[Function[{z, zc}, trialf[z, s, t]], Infinity] /. {z → x + I y, zc → x - I y}},
   Prepend[Select[Flatten[Table[Join[{x, y}, {Re[#], Im[#]} &[mxmy]], {x, -1, 1, 2/n},
       {y, -1, 1, 2/n}], 1], #[[1]]^2 + #[[2]]^2 < 1 &], {"x", "y", "mx", "my"}]];
mzForExport[t_?NumericQ, s_?NumericQ, n_?IntegerQ] :=
  With[{mz = mz[Function[{z, zc}, trialf[z, s, t]], Infinity] /. {z → x + I y, zc → x - I y}},
   Prepend[Select[Flatten[Table[Join[{x, y}, {mz}], {x, -1, 1, 2/n}, {y, -1, 1, 2/n}], 1],
     #[[1]]^2 + #[[2]]^2 < 1 &], {"x", "y", "mz"}]];
exportField[t_?((NumericQ[#] && (Round[# * 100] == # * 100) && (# ≤ 1) && (# ≥ 0)) &),
   s_?NumericQ, n_?IntegerQ] := With[{fname =
     Function[{tstr}, FileNameJoin[{NotebookDirectory[], "..", "figures", "disk_t_" <>
         (StringJoin @@ ToString /@ RotateLeft[Append[RotateRight[IntegerDigits[
             Round[t * 100], 10, 3], 2], "p"], 2]) <> "_" <> tstr <> "_1.TSV"}]]},
   Export[fname["mxmy"], Chop[N[mxmyForExport[t, s, n]]]];
   Export[fname["mz"], Chop[N[mzForExport[t, s, 4 n]]]];
  ];
```

Here the equilibrium values of s are computed for use in exported vector fields. The function minE is defined later in this file.

```
In[•]:= With[{g = 0.1, λ = 0.8}, {minE[0, g, λ], minE[0.1, g, λ], minE[1/2, g, λ], minE[0.6, g, λ],
   minE[0.66, g, λ], minE[0.72, g, λ], minE[0.76, g, λ], minE[0.8, g, λ], minE[1, g, λ]}]
Out[•]=
{{0.0575016, {s → 3.47605}}, {0.0576022, {s → 3.88633}}, {0.0834325, {s → 18.9829}},
 {0.0759393, {s → 7.58385}}, {0.0735066, {s → 7.94141}}, {0.0688854, {s → 4.72547}},
 {0.0709459, {s → 5.10438}}, {0.071585, {s → 4.03626}}, {0.0713789, {s → 3.28118}}}
```

```
In[•]:= exportField[0.1, 3.8863305946522146`, 16];
exportField[0.5, 18.982891418929903`, 16];
exportField[0.6, 7.583847643009591`, 16];
exportField[0.66, 7.941407651200132`, 16];
exportField[0.72, 4.725474799835012`, 16];
exportField[0.76, 5.104380811324221`, 16];
exportField[0.8, 4.036260520089004`, 16];
exportField[1.0, 3.281179556274764`, 16];
```

Let us now compute term by term the total energy (1) in dimensionless units, normalized by $\mu_0 M_S^2 V$ (where the volume of the disk $V = \pi L R^2$).

Exchange energy

```
In[•]:= exchIntegrand = Function[{t, s, r, fi}, Evaluate[FullSimplify[
          (1 / (2 π))(* 1/2 comes from the definition of the exchange stiffness we use C=
             2A, and π is what remains from V after taking out and cancelling LR^2 *)
           sumgradsq[Function[{z, zc}, trialf[z, s, t]], Infinity] /.
            {z → r Exp[I fi], zc → r Exp[-I fi]}, {s > 0, t ≥ 0, t ≤ 1, r > 0, fi > 0, fi < 2 Pi}]]];
      eEX[t_?NumericQ, s_?NumericQ] :=
        Quiet[Re[NIntegrate[Re[exchIntegrand[t, s, r, f]] r,
           {r, 0, 1}, {f, 0, 2 Pi}, Method → "GlobalAdaptive"]],
         {NIntegrate::ncvb, NIntegrate::slwcon, NIntegrate::eincr}];
      (* The GlobalAdaptive strategy produces some warnings,
      which we're going to ignore here,
      even with these warnings the precision
       of the computation is sufficient for the purposes
       of the present manuscript. Alternatively,
      a LocalAdaptive strategy can be used, which is faster,
      produces no warnings,
      but the precision of the result is much worse *)
```

Just few values for testing

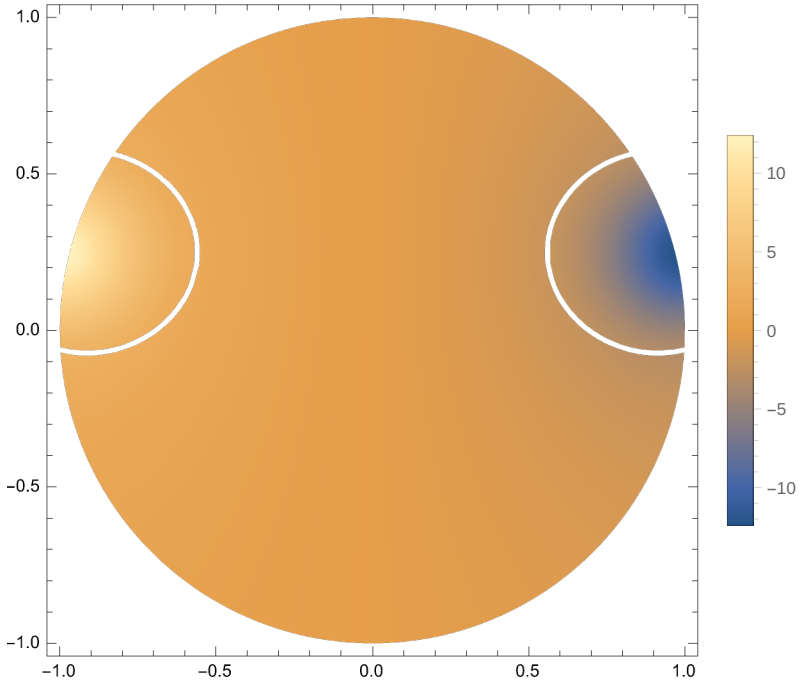
```
In[•]:= {eEX[0, 4], eEX[1 / 2, 4], eEX[1, 4]}
Out[•]= {3.38629, 2.1655, 2.67019}
```

Magnetostatic energy of the volume charges

The charges density looks like this (white artefacts conveniently show the vortex core boundary)

In[ ]:=
```
With[{t = 0.8, s = 4}, Block[{ρ},
  ρ = FullSimplify[
    Re[div[Function[{z, zc}, trialf[z, s, t]], Infinity] /. {z → x + I y, zc → x - I y}]];
  DensityPlot[ρ, {x, -1, 1}, {y, -1, 1}, RegionFunction →
    Function[{x, y, z}, x^2 + y^2 < 1^2], PlotRange → All, PlotLegends → Automatic]]
```

Out[ ]=



In[ ]:=
```
(* reduced 2dx2d integration *)
Clear[invDistanceOverZ]
invDistanceOverZ[b_, g_] = Integrate[1 / Sqrt[b ^ 2 + (z1 - z2) ^ 2],
   {z1, -g / 2, g / 2}, {z2, -g / 2, g / 2}, Assumptions → {b > 0, g > 0}];
Print[invDistanceOverZ[b, g]];
volIntegrand = Function[{t, s, g, r1, fi1, r2, fi2},
   Evaluate[Simplify[(1 / (2 π) (* 1/2 because it is the self-
         energy and 1/π because later normalized to πLR^2*)
      (1 / (4 π) (* the full prefactor for the interaction energy of a pair
        of magnetic charges in SI is μ0/(4π), but μ0 was normalized out *)
      ((Simplify[ComplexExpand[div[Function[{z, zc}, trialf[z, s, t]], Infinity] /.
            {z → r1 Exp[I fi1], zc → r1 Exp[-I fi1]}]])
        (Simplify[ComplexExpand[div[Function[{z, zc}, trialf[z, s, t]], Infinity] /.
            {z → r2 Exp[I fi2], zc → r2 Exp[-I fi2]}]]))
      (invDistanceOverZ[Sqrt[r1 ^ 2 + r2 ^ 2 - 2 r1 r2 Cos[fi1 - fi2]], g]),
     (t ≥ 0) && (t ≤ 1) && (s > 0)]];
(* the argument g=L/R is the aspect ratio of the cylinder *)
eMSvol[t_?NumericQ, s_?NumericQ, g_?NumericQ] :=
 Quiet[
  Re[NIntegrate[Re[volIntegrand[t, s, g, r1, f1, r2, f2]] r1 r2, {r1, 0, 1}, {f1, 0, 2 Pi},
    {r2, 0, r1, 1}, {f2, 0, f1, 2 Pi}, Method → "GlobalAdaptive", MinRecursion → 2]],
  {NIntegrate::ncvb, NIntegrate::slwcon, NIntegrate::eincr, NIntegrate::izero
   (* the  last warning fires when the vortex is centered,
   the volume charges energy integral is truly zero in this case *)}]
```

$$2\,b - 2\,\sqrt{b^2+g^2} - \frac{1}{2}\,g\left(2\,\text{Log}[b] + \text{Log}\left[-g + \sqrt{b^2+g^2}\,\right] - 3\,\text{Log}\left[g + \sqrt{b^2+g^2}\,\right]\right)$$

Just few values for testing

In[ ]:=
```
{eMSvol[0, 4, 0.5], eMSvol[1 / 2, 4, 0.5], eMSvol[1, 4, 0.5]}
```

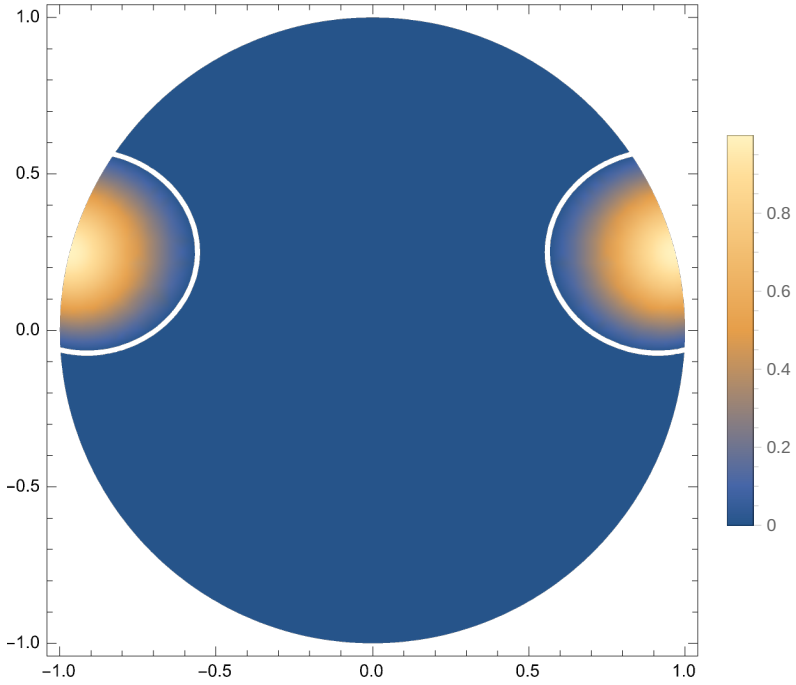
Out[ ]=
```
{0., 0.0189767, 0.0483573}
```

Magnetostatic energy of the face charges

The charges density (on the top face) is (note that unlike the volume charges, the polarity is the same for both half-vortices at the boundary)

In[ ]:=
```
With[{t = 0.8, s = 4}, Block[{σ},
  σ = FullSimplify[
    Re[mz[Function[{z, zc}, trialf[z, s, t]], Infinity] /. {z → x + I y, zc → x - I y}]];
  DensityPlot[σ, {x, -1, 1}, {y, -1, 1}, RegionFunction →
    Function[{x, y, z}, x^2 + y^2 < 1^2], PlotRange → All, PlotLegends → Automatic]]
```

Out[ ]=



```
faceIntegrand = With[{s = 4}, Function[{h, t, s, r1, fi1, r2, fi2},
    Evaluate[FullSimplify[(1 / π (* 1/π because normalized to πLR^2*))
      (1 / (4 π)) (* the full prefactor for the interaction energy of a pair
        of magnetic charges in SI is μ0/(4π), but μ0 was normalized out *))
      ((mz[Function[{z, zc}, trialf[z, s, t]], Infinity] /.
          {z → r1 Exp[I fi1], zc → r1 Exp[-I fi1]}) (mz[Function[{z, zc}, trialf[z, s, t]],
          Infinity] /. {z → r2 Exp[I fi2], zc → r2 Exp[-I fi2]})) /
        Sqrt[r1^2 + r2^2 - 2 r1 r2 Cos[fi1 - fi2] + h^2], (t ≥ 0) && (t ≤ 1)]]];
(* The energy of the interaction of two parallel disks with charge
 distributions of the opposite signs at a distance h from each other,
this function is called here U or Usov's function in honor of Nickolay Usov,
who defined a similar interaction function and
 computed it analytically for centered magnetic vortex *)
funU[h_?NumericQ, t_?NumericQ, s_?NumericQ] := Quiet[
   Re[NIntegrate[Re[faceIntegrand[h, t, s, r1, f1, r2, f2]] r1 r2, {r1, 0, 1}, {f1, 0, 2 Pi},
     {r2, 0, r1, 1}, {f2, 0, f1, 2 Pi}, MinRecursion → 2, Method → "GlobalAdaptive"]],
   {NIntegrate::ncvb, NIntegrate::slwcon, NIntegrate::eincr}];
eMSface[t_?NumericQ, s_?NumericQ, g_?NumericQ] := funU[0, t, s] - funU[g, t, s]
 (* 1/2 due to self-energy is already here because
   there are two faces and U is counted only once *);
```

Just few values for testing

```
In[•]:= {eMSface[0, 4, 0.5], eMSface[1/2, 4, 0.5], eMSface[1, 4, 0.5]}
Out[•]=
{0.00101851, 0.023043, 0.000829259}
```

The total energy $e = E \Big/ \left(\mu_0 \gamma_B M_S^2 \pi L R^2\right) = (L_E/R)^2 e_{EX} + (R/L)(e_{vol} + e_{face}) = (g/\lambda)^2 e_{EX} + (e_{vol}(g) + e_{face}(g))/g$ where $\lambda = L/L_E$ and $g = L/R$. The factor $1/\pi$ is already included into the functions e_{ex}, e_{vol}, e_{face} and the factor 1/g before the magnetostatic energy arises because of the division of the dimension-carrying pre-factor R^3 from the volume(face) integration by (LR^2).

```
eTotal[t_?NumericQ, s_?NumericQ, g_?NumericQ, λ_?NumericQ] :=
  (g/λ)^2 eEX[t, s] + (eMSvol[t, s, g] + eMSface[t, s, g])/g;
(* the following function numerically minimizes
 the total energy over the vortex core size s. *)
minE[t_?NumericQ, g_?NumericQ, λ_?NumericQ] :=
  FindMinimum[eTotal[t, s, g, λ], {s, 0.5, 2}(* EvaluationMonitor:→
    Print["Evaluated at s = ",s],StepMonitor:→Print["Step to s = ",s], *)];
(* calls the previous function,
but just outputs the value of the energy at minimum *)
minEnr[t_?NumericQ, g_?NumericQ, λ_?NumericQ] := minE[t, g, λ][[1]];
```

As a check, let us evaluate the vortex core size at a special point of $\lambda_0 = 2.7284$ (note that $\lambda = L/L_E$), where the vortex core radius R_V is equal to the film thickness L [JMMM 343 (2013), pp. 55-59]. At this point

```
In[•]:= minE[0, 0.2, 2.7284]
Out[•]=
{0.0214769, {s → 4.98961}}
```

Noting that for the centered ($t = 0$) vortex $s = R/R_V$ (because the dimensionless complex coordinate $z = X + I\,Y$ in the trial function is normalized by the cylinder radius) we can express $R_V/L_E = \lambda \Big/ \left(s\,g\right)$

Computing this combination yields R_V/L_E

```
In[•]:= 2.7284/(4.989614557928182` * 0.2)
Out[•]=
2.73408
```

Which ideally must coincide with $\lambda_0 = 2.7284$ we punched in from the start. Note that high numerical precision is not a goal of the present computation, there are much more precise and efficient numerical approaches to address the problem of vortex core size.

The energy of such a particle can also be evaluated from the equation (32) in [JMMM 343 (2013), pp. 55-59]

```
In[•]:= (Log[ρ/λ0] + b0)/ρ^2 /. ρ → λ/g /. {λ0 → 2.7284, b0 → 2.387556, g → 0.2, λ → 2.7284}
Out[•]=
0.0214772
```

which coincides with the above-obtained numerical value to a higher precision .

Now we have all the tools to estimate the energy barrier for vortex split/merger .

Computing each of these three values gives a quick estimate for the barrier height at a particular geometry :

In[•]:=
```
With[{g = 0.1, λ = 1}, {minE[0, g, λ], minE[1/2, g, λ], minE[1, g, λ]}]
```

Out[•]=
```
{{0.0387293, {s → 4.24135}}, {0.0627467, {s → 22.4934}}, {0.0575377, {s → 3.61716}}}
```

In[•]:=
```
With[{g = 0.1, λ = 0.9}, {minE[0, g, λ], minE[1/2, g, λ], minE[1, g, λ]}]
```

Out[•]=
```
{{0.0466961, {s → 3.86176}}, {0.0715295, {s → 19.3528}}, {0.0633486, {s → 3.33054}}}
```

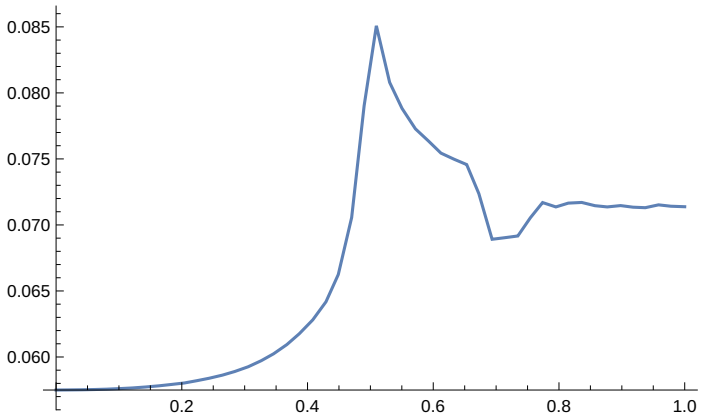
In[•]:=
```
With[{g = 0.1, λ = 0.8}, {minE[0, g, λ], minE[1/2, g, λ], minE[1, g, λ]}]
```

Out[•]=
```
{{0.0575016, {s → 3.47605}}, {0.0834325, {s → 18.9829}}, {0.0713789, {s → 3.28118}}}
```

The plot gives a more detailed profile, but takes a long time to compute (especially with MaxRecursion > 0, due to the numerical scatter).

In[•]:=
```
With[{g = 0.1, λ = 0.8},
 Monitor[Plot[minEnr[t, g, λ], {t, 0, 1}, PlotRange → All, MaxRecursion → 0],
  ProgressIndicator[t, {0, 1}]]]
```

Out[•]=



Some small breaks on the curves are numerical errors due to interplay between the discrete integration lattice and the vortex core boundary moving between its nodes . These errors are visibly small and the existence of the barrier, sampled by many graph points, is beyond any doubt .

The height of the barrier (assuming the cylinder is made of a Permalloy-like material)

In[•]:=
```
0.017 UnitConvert[Quantity["1.0 Teslas"]^2 / Quantity["VacuumPermeability"] Pi
   (0.8 Quantity["5.7 NanoMeters"]) (Quantity["5.7 NanoMeters"] 0.8/0.1)^2, "Joules"]
```

Out[•]=
```
4.0298 × 10^-19 J
```

while seems small is still 2 orders of magnitude above the $k_B T$

In[ ]:=
```
UnitConvert[Quantity["BoltzmannConstant"] × Quantity["300.0 Kelvins"], "Joules"]
```

Out[ ]=

4.14195×10^{-21} J

Note, that this is not an optimized value . For different geometries the barrier can be even higher .

Here we export the computed data for the plot (takes long time).

In[ ]:=
```
dataEmin[g_?NumericQ, λ_?NumericQ, n_?IntegerQ] := Monitor[
  Table[{N[t], Chop[minEnr[t, g, λ]]}, {t, 0, 1, 1/n}], {g, λ, ProgressIndicator[t, {0, 1}]}]
```

In[ ]:=
```
With[{n = 80},
 Export[FileNameJoin[{NotebookDirectory[], "..", "figures", "diskEmin.TSV"}],
   Prepend[Quiet[dataEmin[0.1, 0.8, n]], {"a", "eTot"}]];
]
```