Ilustracja do prawa Ampere'a nr 2

Najpierw liczymy indukcję magnetyczną pochodzącą od obwodu z prądem w kształcie łamanej rozciągającej się między punktami P1 i P2, P2 i P3 oraz P3 i P1.

```
In[1]:= P1 = {-1, -3, -12};

In[2]:= P2 = {1, 2, 10};

In[3]:= P3 = {10, 10, 1};

In[4]:= rp1 = P1 + (P2 - P1) * t
Out[4]= {-1 + 2 t, -3 + 5 t, -12 + 22 t}

In[5]:= s = Solve[rp1[[3]] == 0, t]
```

Out[5]= $\left\{\left\{t \to \frac{6}{11}\right\}\right\}$

```
In[6]:= rp1z0 = rp1 /. s[[1]]
```

Out[6]= $\left\{\frac{1}{11}, -\frac{3}{11}, 0\right\}$

```
In[38]:= Rmin = N[Norm[rp1z0]]
Out[38]= 0.28748

In[8]:= rp2 = P2 + (P3 - P2) * t
Out[8]= {1 + 9 t, 2 + 8 t, 10 - 9 t}

In[9]:= rp3 = P3 + (P1 - P3) * t
Out[9]= {10 - 11 t, 10 - 13 t, 1 - 13 t}

In[10]:=

In[11]:= plot1 = ParametricPlot3D[{rp1, rp2, rp3}, {t, 0, 1}];

In[12]:= r = {x, y, z}
Out[12]= {x, y, z}

In[13]:= d1 = Simplify[Sqrt[(r - rp1).(r - rp1)]];

In[14]:= dl1podt = D[rp1, t];

In[15]:= v1 = Simplify[Cross[dl1podt, r - rp1] / d1^3];

In[16]:= d2 = Simplify[Sqrt[(r - rp2).(r - rp2)]];

In[17]:= dl2podt = D[rp2, t];

In[18]:= v2 = Simplify[Cross[dl2podt, r - rp2] / d2^3];

In[19]:= d3 = Simplify[Sqrt[(r - rp3).(r - rp3)]];

In[20]:= dl3podt = D[rp3, t];
```

```
In[21]:= v3 = Simplify[Cross[dl3podt, r - rp3] / d3^3];

In[22]:= v = Simplify[v1 + v2 + v3];

In[24]:= (* ft=Simplify[Integrate[v,t],
           Element[t,Reals]&&Element[x,Reals]&&Element[y,Reals]&&Element[z,Reals]];*)

In[25]:= ft = Integrate[v, t];

In[26]:= ft1 = Simplify[ft /. {t → 1}];

In[27]:= ft0 = Simplify[ft /. {t → 0}];

In[28]:= B = Simplify[ft1 - ft0];
```

Krążenie uprzednio policzonego pola B liczymy po okręgu o promieniu R=1, środku w początku układu współrzędnych i leżącym w płaszczyźnie z=0

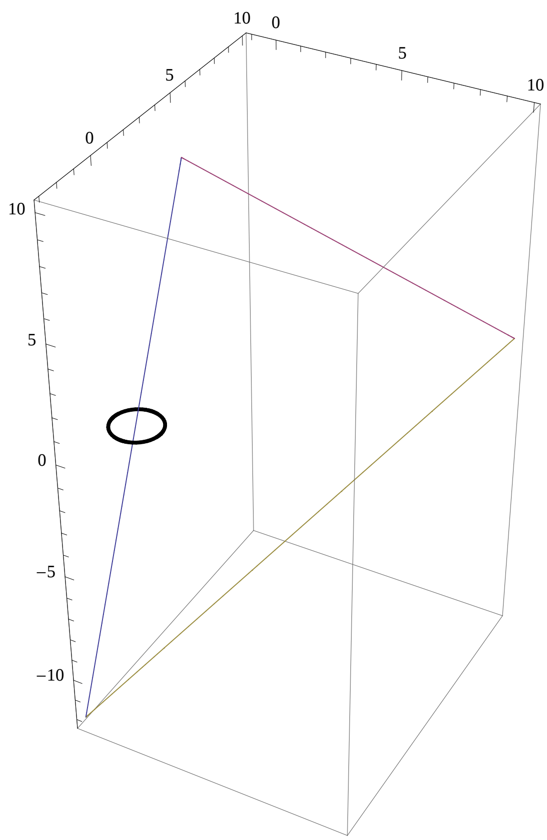
```
In[29]:= rp = {R * Cos[t], R * Sin[t], 0}

Out[29]= {R Cos[t], R Sin[t], 0}

In[30]:= plot2 = ParametricPlot3D[rp /. {R → 1},
           {t, 0, 2 * Pi}, AxesLabel → {"x", "y", "z"}, PlotStyle → Thick];

In[31]:= Show[plot1, plot2]
```

Out[31]=



```
In[32]:= dlpodt = D[rp, t]
Out[32]= {-R Sin[t], R Cos[t], 0}

In[33]:= Brp = B /. {x → R * Cos[t], y → R * Sin[t], z → 0};

In[34]:= cc[R_] = Simplify[Brp.dlpodt];

In[35]:= i3[R_] := NIntegrate[cc[R], {t, 0, 2 * Pi}, Method → {"GlobalAdaptive",
              Method → {"TrapezoidalRule", "Points" → 5000}, "SingularityHandler" → None},
           WorkingPrecision → 16, PrecisionGoal → 12] / (4 * Pi)

In[36]:= i4[R_] := NIntegrate[cc[R], {t, 0, 2 * Pi}, Method → {"GlobalAdaptive",
              Method → {"TrapezoidalRule", "Points" → 5000}, "SingularityHandler" → None},
           WorkingPrecision → 18, PrecisionGoal → 16] / (4 * Pi)

In[37]:= (* Plot[i3[R],{R,1/1000,1}] *)

In[39]:= (* i3[1/100] *)

In[40]:= i3[2 / 10]
```

NIntegrate::slwcon :
Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration is 0, highly oscillatory integrand, or WorkingPrecision too small. ≫

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after 9 recursive bisections in t near {t} = {5.301437602932776}. NIntegrate obtained $-1.404991572258396\times10^{-7}$ and 0.0000138671183904979916. for the integral and error estimates. ≫

Out[40]= $-1.118056768636888\times10^{-8}$

```
In[41]:= i3[1 / 2]
Out[41]= 1.000000000000000

i3[1]

1.000000000000000

i3[20]
```

NIntegrate::slwcon :
Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration is 0, highly oscillatory integrand, or WorkingPrecision too small. ≫

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after 9 recursive bisections in t near {t} = {6.283185307179586}. NIntegrate obtained 7.338715984389646`16.*^-7 and 5.662073613339594`16.*^-7 for the integral and error estimates. ≫

$5.839964624315584\times10^{-8}$

In[42]:=
```
Plot[i3[R], {R, 2 / 10, 20}, PlotPoints → 10, PlotRange → {-1 / 7, 8 / 7}]
```

NIntegrate::slwcon :
Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration is 0, highly oscillatory integrand, or WorkingPrecision too small. ≫

NIntegrate::ncvb : NIntegrate failed to converge to prescribed accuracy after 9 recursive bisections in t near {t} = {5.301437602932776}. NIntegrate obtained 4.245698123225925`16.*^-8 and 0.0000144955381737965`16. for the integral and error estimates. ≫

Out[42]=