Dirt-bags and breezeblocks

Given a width A, a height B, and cuts at a variable angle θ through the centre-point of the block ($x = A/2, y = B/2$, and $\theta = 0$ corresponds to a horizontal cut), the length L of the cut (proportional to the amount of dust) is given by:

$$L/2 = \sqrt{a^2 + b^2},$$

with $a = \frac{A}{2}$ and $b = \frac{A}{2}\tan(\theta)$, or, rearrangingly becoming:

$$L = A\sqrt{1 + \tan(\theta)^2},$$

but this only applies up until θ_{diag} which represents the diagonal from the corners of the block, at which point the line must start to get shorter! This angle is given by

$$\theta_{diag} = \arctan R$$

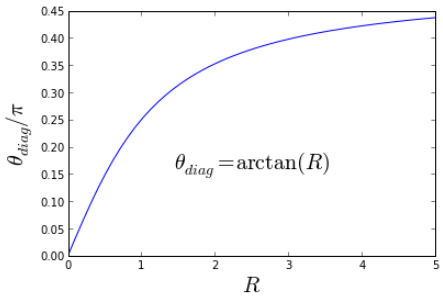
For a ratio of the sides of the block $R = \frac{B}{A}$:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
plt.rcParams['figure.figsize'] = 12, 6
```

In [2]:
```
def f(r):
    return np.arctan(r)
R_max = 5.0
R = np.linspace(0.0, R_max, 50)
theta_max = f(R)
```

In [3]:
```
figure(figsize = (6,4))
plt.plot(R, theta_max/pi)
plt.text(0.5*R_max, 0.5/pi,r"$\theta_{diag} = \arctan(R)$", horizontalalignment='center', fontsize=20);
plt.xlabel(r"$R$", fontsize = 20)
plt.ylabel(r"$\theta_{diag}/\pi$", fontsize = 20)
```

Out[3]: <matplotlib.text.Text at 0x3bea150>



Which, as we would expect, approaches $\theta_{diag} = \pi/2$ (or 90°) as the ratio $R = B/A$ gets larger and approaches infinity. The easiest way to see this is if you imagine the block being extremely tall or extremely narrow - the diagonal is essentially in line with the long axis of the block!

Regardless, once we reach θ_{diag}, the formula for the length of the cut changes:

$$L/2 = \sqrt{\alpha^2 + \beta^2},$$

with $\alpha = \frac{B}{2}\tan(\phi)$,

$$\phi = \pi/2 - \theta,$$

and $\beta = \frac{B}{2}$. Or:

$$L = B\sqrt{\tan(\phi)^2 + 1}.$$

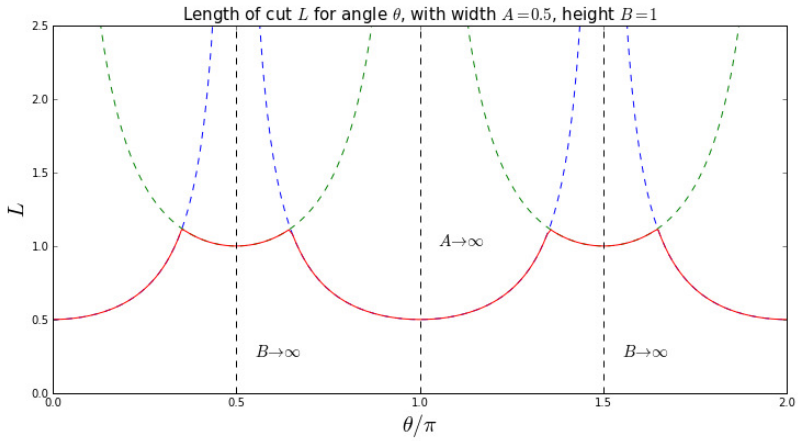
So we put this all together and plot the length of the cut as a function of the angle of the cut:

(NB the blue and green dashed lines show the length of the cut if θ_{diag} was very large i.e. $B, A \to \infty$)

In [23]:
```
def g(t):
    return np.sqrt(np.tan(t)**2 + 1)
A = 0.5
B = 1
R = B/A
t_max = f(R)

points = 200
t = np.linspace(0,2*pi, points)
LA = A*g(t)
p = pi/2-t
LB = B*g(p)

t_mod = [pi/2-x if (pi-t_max > x > t_max) or (2*pi - t_max > x > pi+t_max) else x for x in t]
L = [B*g(x) if (pi-t_max > y > t_max) or (2*pi - t_max > y > pi+t_max) else A*g(x) for x,y in zip(t_mod,t)]
```

In [26]:
```
plt.plot(t/pi, LA, 'b--')
plt.plot(t/pi, LB, 'g--')

plt.plot(t/pi, L, 'r')

#plt.text(0.35*t_max/pi, 1.5,r"$L/A = \sqrt{\tan(\theta)^2 + 1}$", horizontalalignment='center', fontsize=20);
plt.xlabel(r"$\theta/\pi$", fontsize = 20)
plt.ylabel(r"$L$", fontsize = 20)
plt.ylim(ymax = 2.5, ymin = 0)

plt.axvline(x = 0.5, color = 'k', linestyle = '--')
plt.text(0.55,0.25,r"$B \to \infty$", horizontalalignment = 'left', fontsize = 15)

plt.axvline(x = 1.5, color = 'k', linestyle = '--')
plt.text(1.55,0.25,r"$B \to \infty$", horizontalalignment = 'left', fontsize = 15)

plt.axvline(x = 1, color = 'k', linestyle = '--')
plt.text(1.05,1,r"$A \to \infty$", horizontalalignment = 'left', fontsize = 15)

plt.title(r"Length of cut $L$ for angle $\theta$, with width $A = 0.5$, height $B = 1$", fontsize = 15)
```

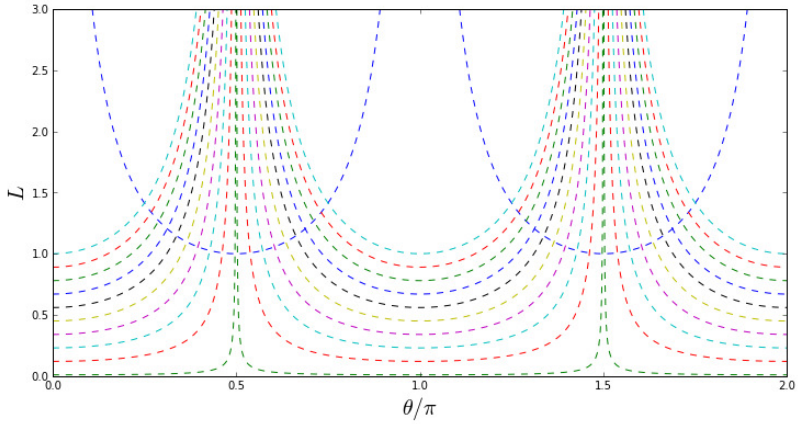
Out[26]: <matplotlib.text.Text at 0x5edbed0>



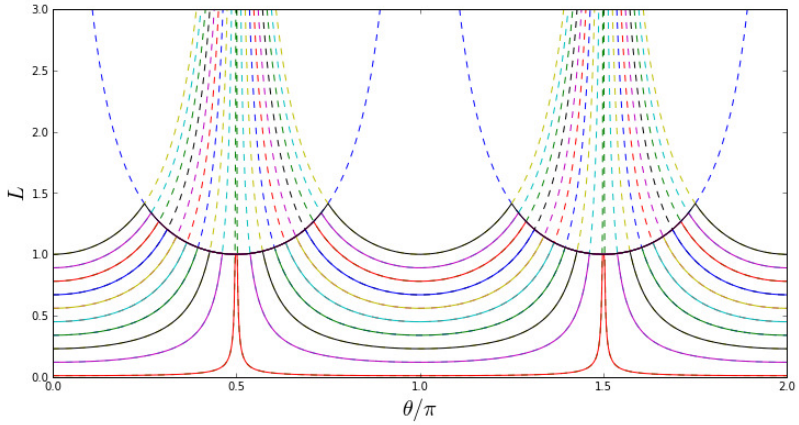
The red line shows the length of the cut, which is directly proportional to the amount of dust created. Next, more or less for giggles, we plot the same graph for a range of different A s, with a fixed B.

In [6]:
```
N = 10
A_ = np.linspace(0.01,1,N)
B = 1
R = B/A_
t_max_ = f(R)
variables = [{'t_max':var[0], 'A':var[1]} for var in zip(t_max_,A_)]
```

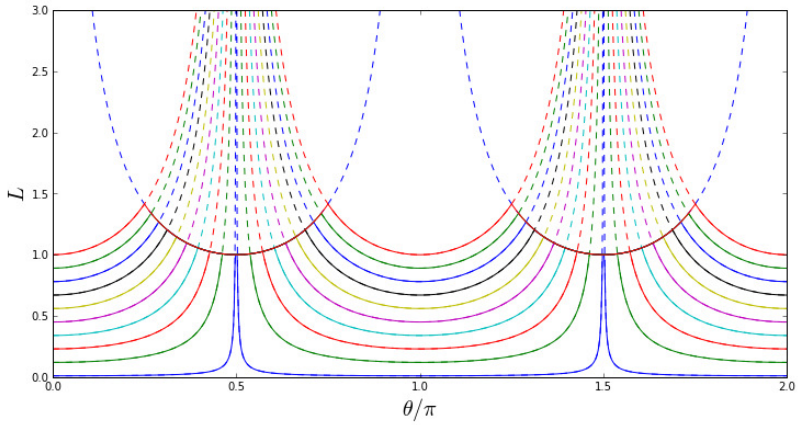
In [7]:
```
points = 1000
t = np.linspace(0,2*pi,points)
p = pi/2 - t
LB = B*g(p)
plt.plot(t/pi,LB, '--')
plt.ylim(ymax = 3, ymin = 0)
plt.xlim(xmax = 2)
plt.xlabel(r"$\theta/\pi$", fontsize = 20)
plt.ylabel(r"$L$", fontsize = 20)
for v in variables:
    LA = v['A']*g(t)
    plt.plot(t/pi,LA, '--')
```



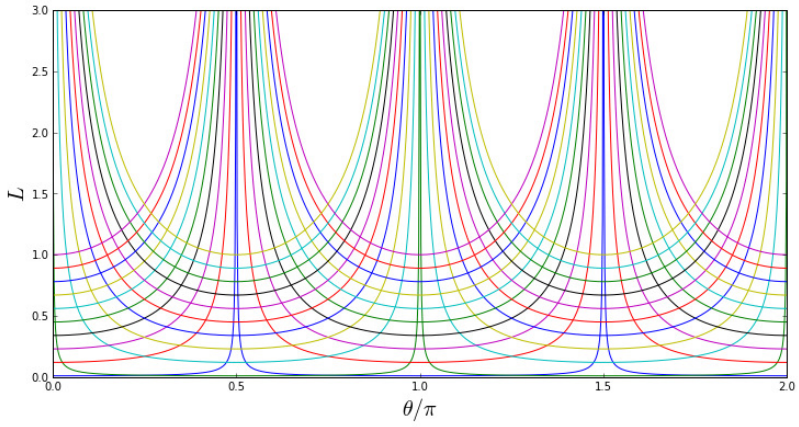
In [8]:
```
points = 1000
t = np.linspace(0,2*pi,points)
p = pi/2 - t
LB = B*g(p)
plt.plot(t/pi,LB, '--')
plt.ylim(ymax = 3, ymin = 0)
plt.xlim(xmax = 2)
plt.xlabel(r"$\theta/\pi$", fontsize = 20)
plt.ylabel(r"$L$", fontsize = 20)
for v in variables:
    LA = v['A']*g(t)
    plt.plot(t/pi,LA, '--')
    t_mod = [pi/2-x if ((pi-v['t_max']) > x and x > v['t_max']) or ((2*pi - v['t_max']) > x and x > (pi+v['t_max'])) else x for x in t]
    L = [B*g(x) if (pi-v['t_max'] > y > v['t_max']) or (2*pi - v['t_max'] > y > pi+v['t_max']) else v['A']*g(x) for x,y in zip(t_mod,t)]
    plt.plot(t/pi,L)
```



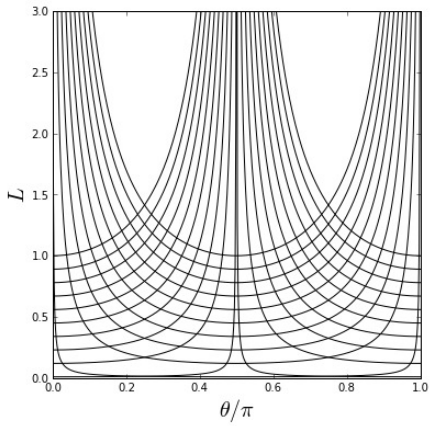
In [9]:
```
colors = ['b','g','r','c','m','y','k','b','g','r']
variables = [{'t_max':var[0], 'A':var[1], 'c':var[2]} for var in zip(t_max_,A_,colors)]
points = 1000
t = np.linspace(0,2*pi,points)
p = pi/2 - t
LB = B*g(p)
plt.plot(t/pi,LB, '--')
plt.ylim(ymax = 3, ymin = 0)
plt.xlim(xmax = 2)
plt.xlabel(r"$\theta/\pi$", fontsize = 20)
plt.ylabel(r"$L$", fontsize = 20)
for v in variables:
    LA = v['A']*g(t)
    plt.plot(t/pi,LA, '--', color = v['c'])
    t_mod = [pi/2-x if ((pi-v['t_max']) > x and x > v['t_max']) or ((2*pi - v['t_max']) > x and x > (pi+v['t_max'])) else x for x in t]
    L = [B*g(x) if (pi-v['t_max'] > y > v['t_max']) or (2*pi - v['t_max'] > y > pi+v['t_max']) else v['A']*g(x) for x,y in zip(t_mod,t)]
    plt.plot(t/pi,L,color = v['c'])
```



In [10]:
```
points = 1000
t = np.linspace(0,2*pi,points)
p = pi/2 - t
#LB = B*g(p)
#plt.plot(t/pi,LB, '--')
plt.ylim(ymax = 3, ymin = 0)
plt.xlim(xmax = 2)
plt.xlabel(r"$\theta/\pi$", fontsize = 20)
plt.ylabel(r"$L$", fontsize = 20)
for v in variables:
    LA = v['A']*g(t)
    LB = v['A']*g(p)
    plt.plot(t/pi,LA, '-')
    plt.plot(t/pi,LB, '-')
```



In [11]:
```
points = 1000
t = np.linspace(0,2*pi,points)
p = pi/2 - t
#LB = B*g(p)
#plt.plot(t/pi,LB, '--')
figure(figsize = (6,6))
ylim(ymax = 3, ymin = 0)
xlim(xmax = 1)
xlabel(r"$\theta/\pi$", fontsize = 20)
ylabel(r"$L$", fontsize = 20)
for v in variables:
    LA = v['A']*g(t)
    LB = v['A']*g(p)
    plot(t/pi,LA, 'k-')
    plot(t/pi,LB, 'k-')
```



Now given that $\theta_{max} = \arctan(R)$,

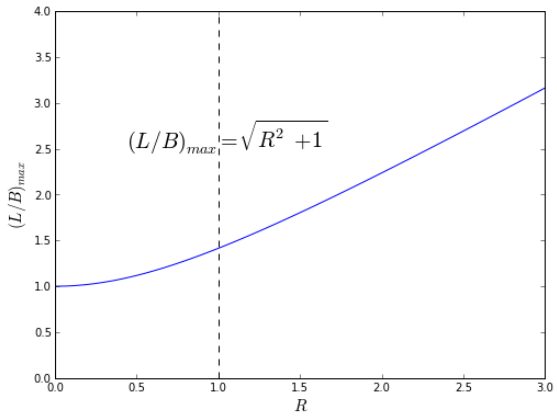
$(L/B)_{max} = \sqrt{R^2 + 1}$.

So for instance if the block has a square face, then $R = B/A = 1$, and the proportion of the smallest dust pile to the biggest is $\sqrt{2} \sim 1.4$. or if it's twice as long as it is tall, then the ratio is $\sqrt{6} \sim 2.4$.

In [12]:
```
def h(r):
    return np.sqrt(r**2 + 1)
R_max = 3
R = np.linspace(0, R_max, 50)
L = h(R)
```

In [13]:
```
figure(figsize = (8,6))
plt.plot(R, L)
plt.text(0.35*R_max, 2.5,r"$(L/B)_{max} = \sqrt{R^2 + 1}$", horizontalalignment='center', fontsize=20);
plt.ylim(ymin = 0, ymax = 4)
plt.xlabel(r"$R$", fontsize = 15)
plt.ylabel(r"$(L/B)_{max}$", fontsize = 15)
plt.axvline(x = 1.0, color = 'k', linestyle = '--')
```

Out[13]: <matplotlib.lines.Line2D at 0x3e52710>



But what about the derivative? The operator $f'(x)$ is defined as the difference quotient in the limit that $h \to 0$

$f'(x) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$,

or in the more common notation: $\frac{dy}{dx}$. This second notation is sometimes said to be misleading - suggesting the treatment of the symbols as the separate components of a regular fraction.

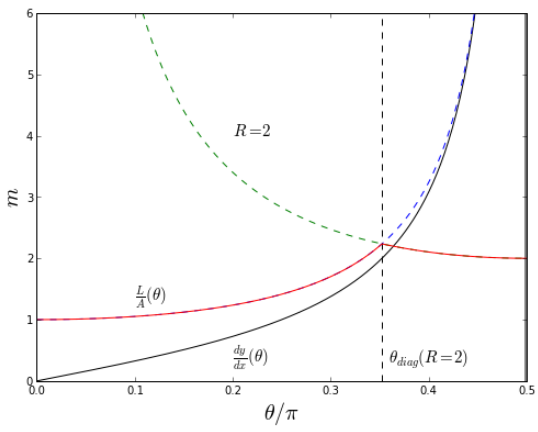
The gradient of a straight line with angle θ from the x-axis is:

$m = \tan(\theta)$,

seen against our previous plot (where we divide L by A to make our approximation of the slope dimensionless):

In [27]:
```
points = 1000
t = np.linspace(0,2*pi,points)
p = pi/2 - t
figure(figsize = (8,6))
plot(t/pi,np.tan(t),'k',linewidth = 1)
ylim(ymax = 6, ymin = 0)
xlim(xmax = 0.5)
xlabel(r"$\theta/\pi$", fontsize = 20)
ylabel(r"$m$", fontsize = 20)
text(0.1,1.3,r"$\frac{L}{A}(\theta)$", horizontalalignment = 'left', fontsize = 15)
text(0.2,0.3,r"$\frac{dy}{dx}(\theta)$", horizontalalignment = 'left', fontsize = 15)
R=2
LA = g(t)
plot(t/pi,LA, '--')
LB = g(p)*R
plot(t/pi,LB, '--')
cutoff = np.arctan(R)
axvline(x = cutoff/pi, color = 'k', linestyle = '--')
text(0.2,4,r"$R=2$",fontsize=15)
text(1.02*cutoff/pi, 0.3, r"$\theta_{diag}(R=2)$",horizontalalignment = 'left', fontsize = 15)

t_mod = [pi/2-x if (pi-t_max > x > t_max) or (2*pi - t_max > x > pi+t_max) else x for x in t]
L = [g(x)*R if (pi-t_max > y > t_max) or (2*pi - t_max > y > pi+t_max) else g(x) for x,y in zip(t_mod,t)]
plot(t/pi,L)
```

Out[27]: [<matplotlib.lines.Line2D at 0x5cba6d0>]



From before, we have:

$L/A = \sqrt{1 + \tan(\theta)^2}$.

Clearly we can see a discrepancy between L and the gradient of the line - we have an errant 1 inside the square root!

In order to get the gradient from L (and thus from the amount of dust we generate from the cut) we must rearrange:

$$m = \tan(\theta) = \sqrt{(\frac{L}{A})^2 - 1}$$

This shows us more clearly that the length of the cut L is only approximately proportional to the gradient m when $L \gg A$,

but since $L \propto A$ this can only be achieved at large $\tan(\theta)$ i.e. $\theta \sim \pi/2$ i.e. $B \gg A$.