Example for Convolution

In [1]:
```
] add DSP
```

```
   Updating registry at `~/.julia/registries/General`
################################################################################
## 100.0%
  Resolving package versions...
No Changes to `~/.julia/environments/v1.5/Project.toml`
No Changes to `~/.julia/environments/v1.5/Manifest.toml`
```

In [2]:
```
using DSP
```

$$p_1(x) = 3 + 5x + x^2$$

In [3]:
```
a=[3,5,1]
```

Out[3]:
```
3-element Array{Int64,1}:
 3
 5
 1
```

$$p_2(x) = 1 - 2x + 3x^2 - 5x^3$$

In [7]:
```
b=[1,-2,3,-5]
```

Out[7]:
```
4-element Array{Int64,1}:
  1
 -2
  3
 -5
```

In [8]:
```
conv(a,b)
```

Out[8]:
```
6-element Array{Int64,1}:
   3
  -1
   0
  -2
 -22
  -5
```

length of the vector c should be 6

Therefore

$$p_1(x)p_2(x) = 3 - x - 2x^3 - 22x^4 - 5x^5$$

In [9]:
```
f(x)=sqrt(x)*(x/100+sin(x/10))
```

Out[9]:
```
f (generic function with 1 method)
```

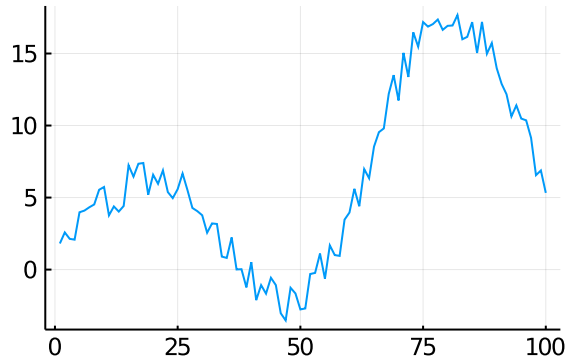
In [11]:
```

```

In [16]:
```
xs=f.(ns)+3*rand(length(ns));
```

In [13]:
```
using Plots
```

In [17]:
```
plot(ns,xs,size=[300,200],legend=false)
```

Out[17]:



In [20]:
```
a=[1/3,1/3,1/3]
```

Out[20]:
```
3-element Array{Float64,1}:
 0.3333333333333333
 0.3333333333333333
 0.3333333333333333
```

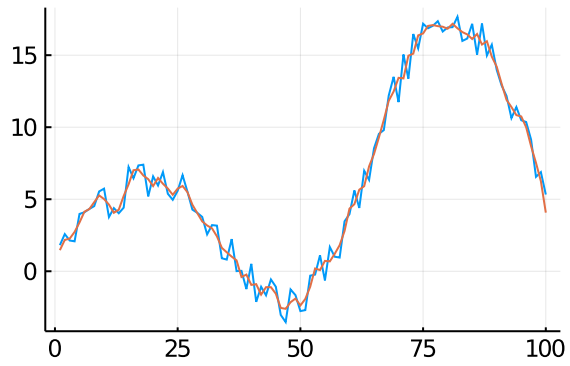
In [21]:
```
cs=conv(a,xs);
```

In [22]:
```
length(cs)
```

Out[22]:
```
102
```

In [23]:
```
plot!(ns,cs[2:101])
```

Out[23]:

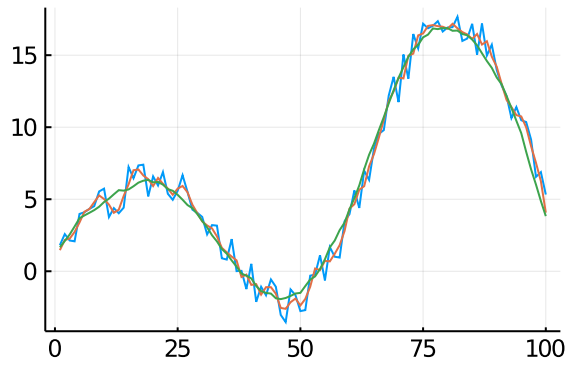


In [27]:
```
anew=ones(11)/11;
```

In [25]:
```
cnew=conv(anew,xs);
```

In [28]:
```
plot!(ns,cnew[6:105])
```

Out[28]:



In [ ]:
```

```