# UniDyn--Demo-02.nb

John A. Marohn
jam99@cornell.edu
Cornell University

**Abstract:** Use the **UniDyn** Evolver function to calculate the evolution of the magnetization of a single spin 1/2 particle under off-resonance, variable-phase irradiation. Plot the evolving magnetization for various combinations of resonance offset and irradiation phase.

## Set the path to the package

Tell *Mathematica* the path to the directory containing the package.

EDIT THE FOLLOWING PATH STRINGS:

In[1]:=
```
$UniDynPath =
  "/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/
    unidyn";
```

YOU SHOULD NOT NEED TO EDIT ANYTHING FROM HERE ONWARDS.

## Load the package

Append the package path to the system path. Before trying to load the package, ask *Mathematica* to find it. This is a test that we directed *Mathematica* to the correct directory. The output of this command should be the full system path to the UniDyn.m file.

In[2]:=
```
$Path = AppendTo[$Path, $UniDynPath];
FindFile["UniDyn`"]
```

Out[3]=
```
/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/unidyn/UniDyn.m
```

Now that we are confident that the path is set correctly, load the package. Setting the global $VerboseLoad variable to True will print out the help strings for key commands in the package.

In[4]:=
```
$VerboseLoad = True;
Needs["UniDyn`"]
```

CreateOperator : CreateOperator [] is used to batch −define a bunch of operators. Example: CreateOperator [{{Ix, Iy, Iz },{Sx,Sy,Sz }}] will create six operators, where each of the operators in the first list will commute with each of the operators of the second list.

CreateScalar : CreateScalar [list ] is used to batch −define a bunch of scalars. The parameter list can be a single scalar or a list of scalars. Example: CreateScalar [{w1,w2 }].

NCSort : NCSort [list ] sorts the operators in list into canonical order.

SortedMult : SortedMult [list ] returns Mult [list$ordered ], where list$ordered are the elements of list sorted into canonical order.

MultSort : MultSort [NonCommutativeMultiplyt [list ]] returns returns NonCommutativeMultiply [list$ordered ], where list$ordered are the elements of list sorted into canonical order.

Comm : Comm [a,b ] calculates the commutator of two operators.

SpinSingle$CreateOperators : SpinSingle$CreateOperators [Ix,Iy,Iz,L ] creates Ix, Iy, and Iz angular momentum operators and defines their commutation relations. When the total angular momentum L = 1/2, additional rules are defined to simplify products of the angular momentum operators. When the total angular momentum L is unspecified, no such simplification rules are defined.

OscSingle$CreateOperators : OscSingle$CreateOperators [aL,aR ] creates a raising operator aR and a lowering operator aL for single harmonic oscillator and defines the operator commutation relations.

Evolve : Evolve [H, t, ρ] represents unitary evolution of the density operator ρ for a time t under the Hamiltonian H. This function expands according to simplification rules but leaves the evolution unevaluated.

Evolver : Evolver [H, t, ρ(0)] calculates ρ(t) = Exp [−I H t ] ρ(0) Exp [+I H t ], assuming that H is time independent, according to the commutation rules followed by ρ(0) and H.

# Function to help draw the magnetization

In[6]:=
```
Clear[my$drawing];
SetAttributes[my$drawing, HoldAll];

my$drawing[func_[t_, a___], t$final_, N$step_] :=

 Module[{ρ$vector$data, ρ$arrows, axes$arrows, big$plot},

  (* Calculate a final time and a time step *)
  (* The final time point should not be included in the plot *)

  T$max = t$final * (N$step - 1) / N$step;
```

```
T$step = t$final / N$step;

(* Make a table of data of the form *)
(* {{0.,{0.,0.,1.}},{0.5,{0.,-0.9,-0.5}} *)

ρ$vector$data =
 Table[{N[t / T$max], N[func[t, a]]}, {t, 0, T$max, T$step}];

(* Add arrows; the arrows grow from
 light to dark as time progresses in the plot*)

ρ$arrows =
 Graphics3D[{GrayLevel[1.0 - N[#[[1]]]], {Arrowheads[0.015],
      Arrow[Tube[{{0, 0, 0}, #[[2]]}]]}}] & /@ ρ$vector$data;

(* Add axes arrows.  Here we make
 the assumption that the magnetization vector *)
(*  has a magnitude of 1. *)

axes$arrows =
 Graphics3D[{Black, Arrow[Tube[{{0, 0, 0}, #}]]}] & /@
  {{0, 0, 1.25}, {0, 1.25, 0}, {1.25, 0, 0},
   {0, 0, -1.25}, {0, -1.25, 0}, {-1.25, 0, 0}};

big$plot = Flatten[Append[ρ$arrows, axes$arrows]];

(* Add axes labels. *)

big$plot = Flatten[Append[big$plot,
   Graphics3D[Text[Style[z, Large], {0, 0, 1.35}]]]];
big$plot = Flatten[Append[big$plot,
   Graphics3D[Text[Style[y, Large], {0, 1.35, 0}]]]];
big$plot = Flatten[Append[big$plot,
   Graphics3D[Text[Style[x, Large], {1.35, 0, 0}]]]];
```

```
(* Plot all the arrows.  The neutral
 lighting helps making the rendering fast -- *)
(*  the default Mathematica camera has three-
 colored lights which makes funny reflections *)
(*  of off small objects like our arrows. *)

Show[big$plot, Boxed → False, ViewVertical → {0, 0, 1},
 ViewPoint → {2.0, -1.0, 1.0}, Lighting → "Neutral"]

]
```

# Examples of unitary evolution in a spin 1/2 system

## Create a single spin

The assumptions define below are required for *Mathematica* to recognize $\sqrt{-\Delta^2 - \omega^2} = i\sqrt{\Delta^2 + \omega^2}$ inside an exponential.   One of the variables has to be defined to be $> 0$ and not just $\geqslant 0$.

In[9]:=
```
Clear[

 Δ,            (* resonance offset frequency *)
 ω,            (* Rabi frequency of the applied irradiation *)
 ϕ,            (* phase of the applied irradiation *)
 t,            (* time *)
 Ix, Iy, Iz,   (* spin angular momentum operators *)
 ρ,            (* spin density operator *)
 ρ$0,          (* initial spin density operator *)
 H             (* spin Hamiltonian *)]

CreateScalar[Δ, ω, ϕ , t];
SpinSingle$CreateOperators[Ix, Iy, Iz, L = 1 / 2];

$Assumptions = {Element[Δ, Reals], Δ ≥ 0,
   Element[ω, Reals], ω > 0, Element[t, Reals], t ≥ 0};
```

SpinSingle$CreateOperators : Creating spin operators.

SpinSingle$CreateOperators : Adding spin commutations relations.

SpinSingle$CreateOperators : Angular momentum L = 1/2. Adding operator simplification rules.

## Off-resonance variable-phase nutation

Irradiation Hamiltonian written in the interaction representation. The intial density operator is parallel to $I_z$.

In[13]:=
```
H = Δ Iz + ω (Cos[ϕ] Ix + Sin[ϕ] Iy);
ρ$0 = Iz;
```

Calculating the time-dependent density operator might take as long as 10 to 15 seconds to complete.

In[15]:=
```
ρ [t_, Δ_, ω_, ϕ_] = Collect[
       (Evolver[H, t, ρ$0] // Simplify // ExpToTrig //
     FullSimplify),
       {Ix, Iy, Iz}];
```

In[16]:=
```
ρ[t, Δ, ω, ϕ] /. {Δ → Subscript[ω, 0], ω → Subscript[ω, 1]}
```

Out[16]=
$$\frac{\text{Iz}\left(\omega_0^2+\text{Cos}\left[t\sqrt{\omega_0^2+\omega_1^2}\right]\omega_1^2\right)}{\omega_0^2+\omega_1^2}+\frac{\text{Iy}\left(\text{Sin}[\phi]\,\omega_0\,\omega_1-\text{Cos}\left[t\sqrt{\omega_0^2+\omega_1^2}\right]\text{Sin}[\phi]\,\omega_0\,\omega_1-\text{Cos}[\phi]\,\text{Sin}\left[t\sqrt{\omega_0^2+\omega_1^2}\right]\omega_1\sqrt{\omega_0^2+\omega_1^2}\right)}{\omega_0^2+\omega_1^2}+\frac{\text{Ix}\left(\text{Cos}[\phi]\,\omega_0\,\omega_1-\text{Cos}[\phi]\,\text{Cos}\left[t\sqrt{\omega_0^2+\omega_1^2}\right]\omega_0\,\omega_1+\text{Sin}[\phi]\,\text{Sin}\left[t\sqrt{\omega_0^2+\omega_1^2}\right]\omega_1\sqrt{\omega_0^2+\omega_1^2}\right)}{\omega_0^2+\omega_1^2}$$

Below we want a function that returns a triple of numbers describing the magnetization vector. We turn the above expression for the density operator into a triple of numbers using the *Mathematica* function Coefficient. I tried using the NCAlgebra's NCCoefficient function but could not get it to work. The function below does what we want.

In[17]:=
```
ρ$vector[t_, Δ_, ω_, ϕ_] =
  Simplify[ Coefficient[ρ[t, Δ, ω, ϕ], #, 1] & /@ {Ix , Iy, Iz}]
```

Out[17]=
$$\left\{\frac{\omega\left(\text{Cos}[\phi]\left(\Delta-\Delta\,\text{Cos}\left[t\sqrt{\Delta^2+\omega^2}\right]\right)+\sqrt{\Delta^2+\omega^2}\,\text{Sin}[\phi]\,\text{Sin}\left[t\sqrt{\Delta^2+\omega^2}\right]\right)}{\Delta^2+\omega^2},\ -\frac{\omega\left(\Delta\left(-1+\text{Cos}\left[t\sqrt{\Delta^2+\omega^2}\right]\right)\text{Sin}[\phi]+\sqrt{\Delta^2+\omega^2}\,\text{Cos}[\phi]\,\text{Sin}\left[t\sqrt{\Delta^2+\omega^2}\right]\right)}{\Delta^2+\omega^2},\ \frac{\Delta^2+\omega^2\,\text{Cos}\left[t\sqrt{\Delta^2+\omega^2}\right]}{\Delta^2+\omega^2}\right\}$$

## Check limiting cases

On resonance, the effective field is in the x-y plane. The z magnetization will oscillate co-sinusoidally while the magnetization in the x-y plane will oscillate sinusoidally.

In[18]:=
```
ρ[t, 0, ω, ϕ] // PowerExpand // FullSimplify
```

Out[18]= $\text{Iz}\,\text{Cos}[t\,\omega]+(-\text{Iy}\,\text{Cos}[\phi]+\text{Ix}\,\text{Sin}[\phi])\,\text{Sin}[t\,\omega]$

Apply a $\pi$ pulse. Observe that the magnetization is indeed inverted.

In[19]:=
```
ρ[ π / ω, 0, ω, 0] // PowerExpand
```

Out[19]= $-\text{Iz}$

Now apply a $\pi/2$ pulse. Appying an “x” pulse, one with a relative phase of $\phi$=0, places the magnetization along the -y axis. A “y” pulse, one with a relative phase of $\phi = \pi/2$, places the magnetization along the +x axis.

In[20]:=
```
ρ[π / (2 ω), 0, ω, #] & /@ {0, π / 2} // PowerExpand
```

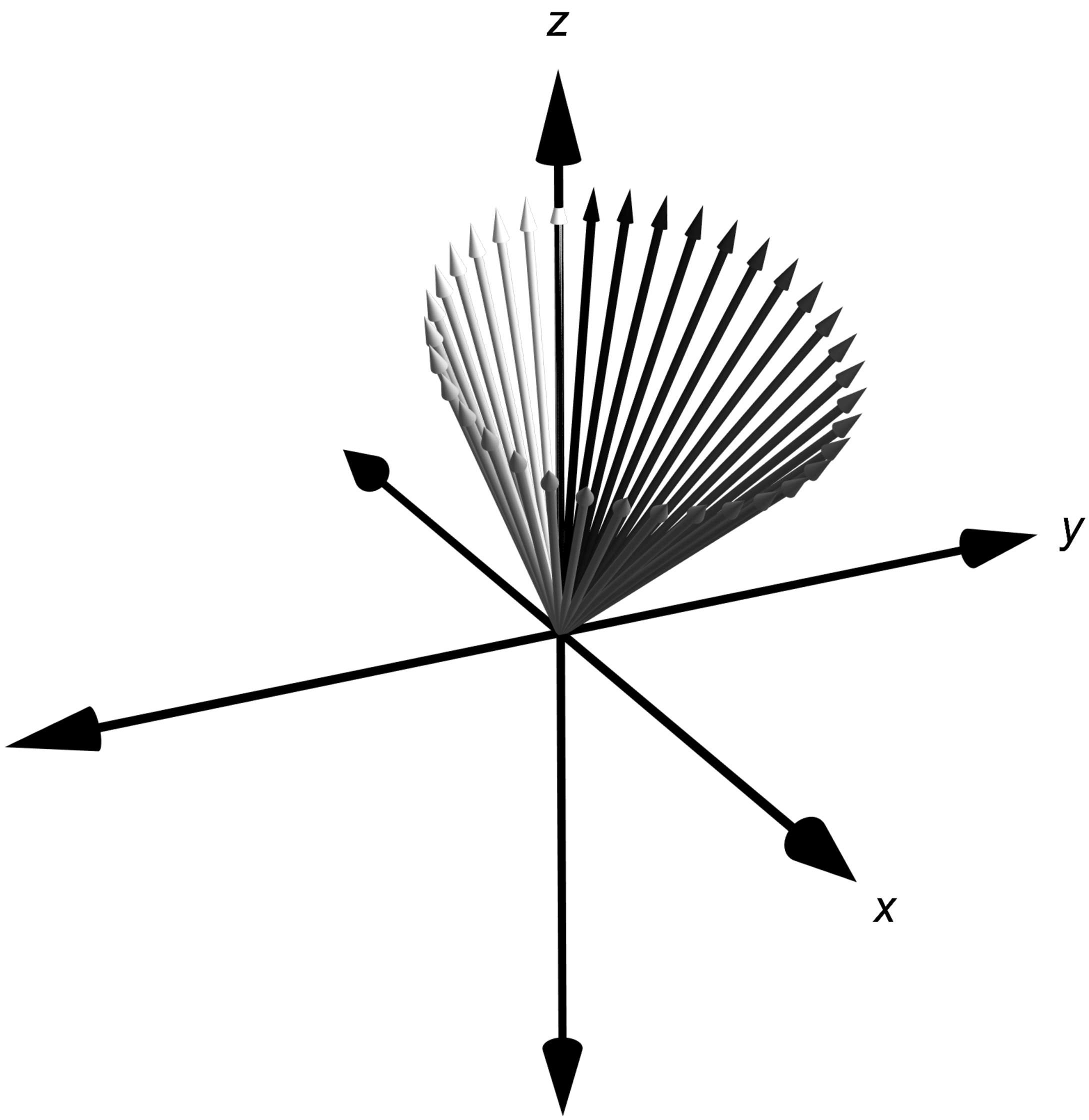
Out[20]=
```
{-Iy, Ix}
```

## Draw the magnetization

Set the rf phase to $\phi = 0$, set the Rabi frequency to $\omega = 1$, and set the resonance offset to $\Delta = 2$. The effective field has a magnitude of $\sqrt{2^2 + 1^2} = \sqrt{5}$, so we’ll watch the magnetization out to a time of $2\pi \big/ \sqrt{5}$ in order to capture the magnetization orbiting once around the effective field.

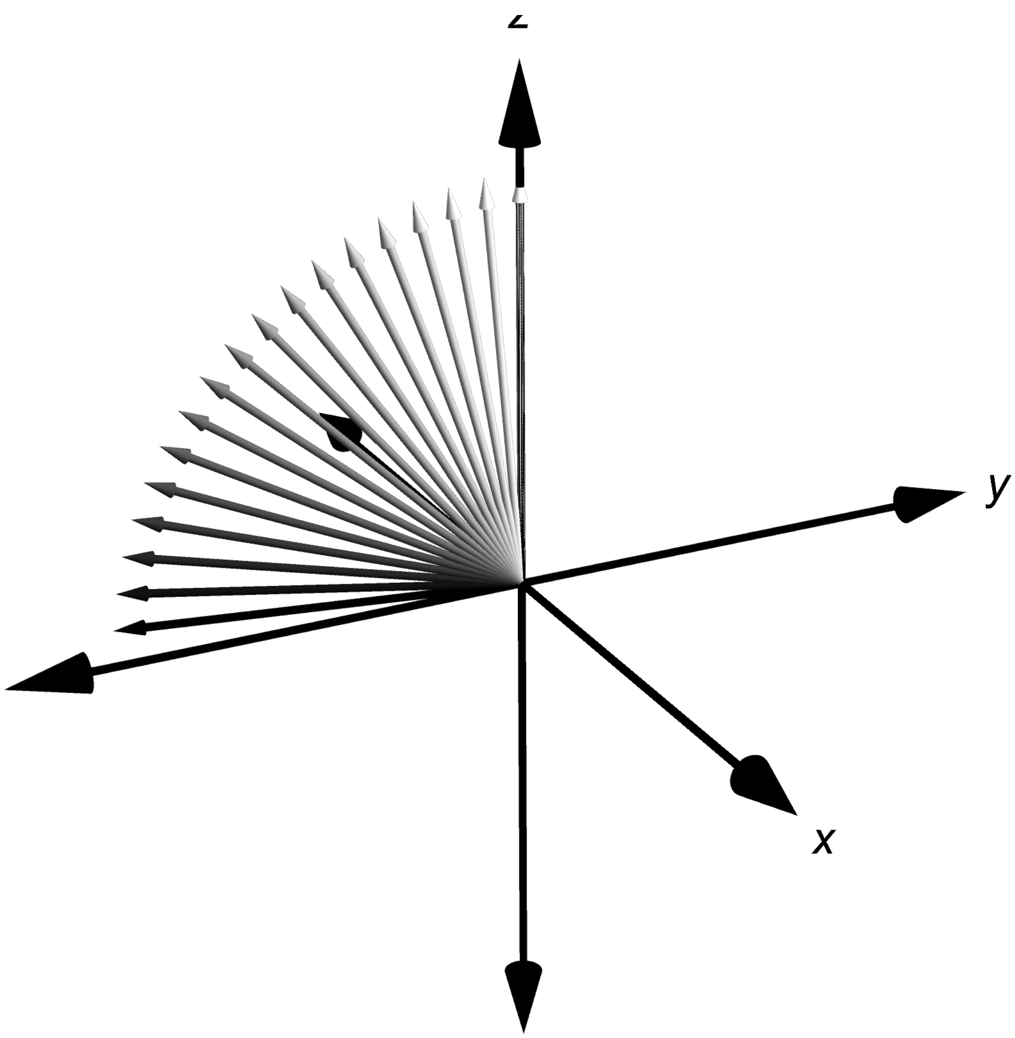
In[21]:=
```
Show[my$drawing[ρ$vector[t, 2, 1, 0], 2 π/√5 , 36], ImageSize → Full]
```

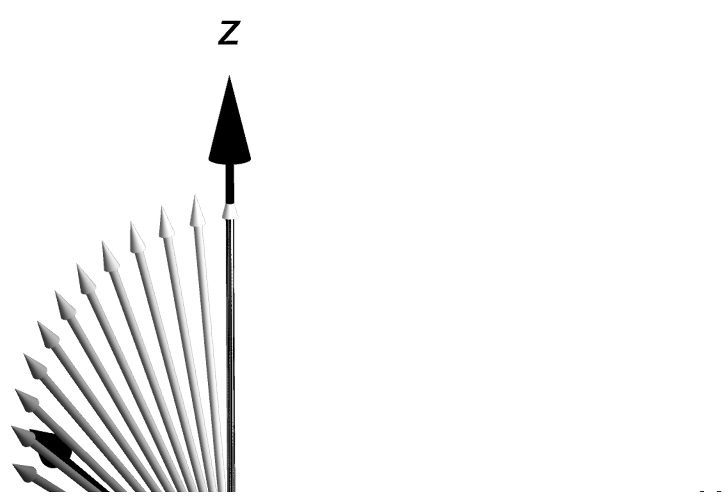
Out[21]=

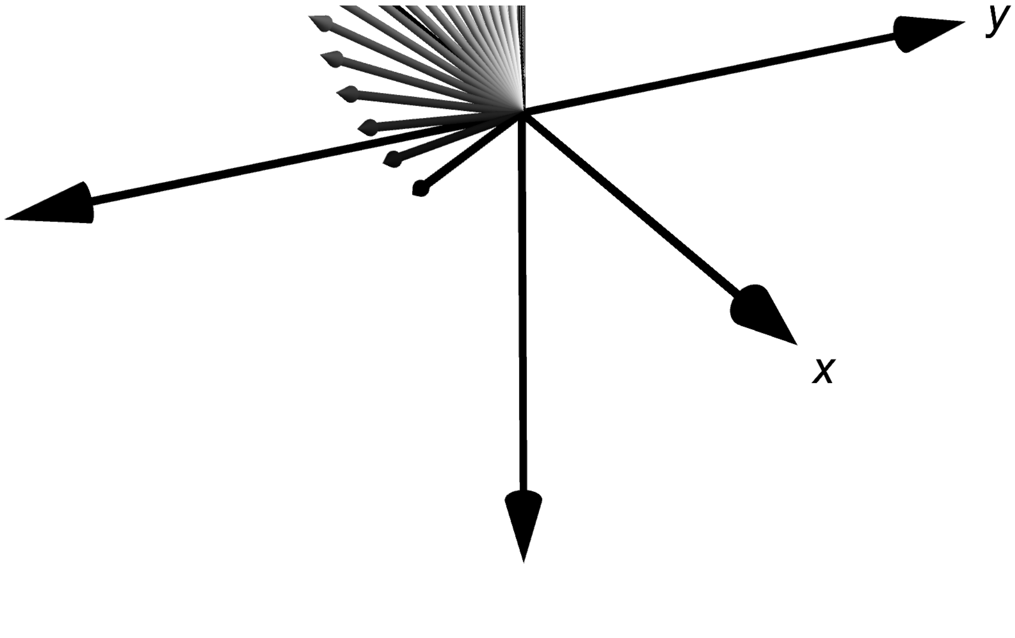



Set the rf phase to $\phi = 0$, set the Rabi frequency to $\omega = 1$, and look at magnetization out to times equal to $\pi/2$. Vary the resonance offset and plot the magnetization.

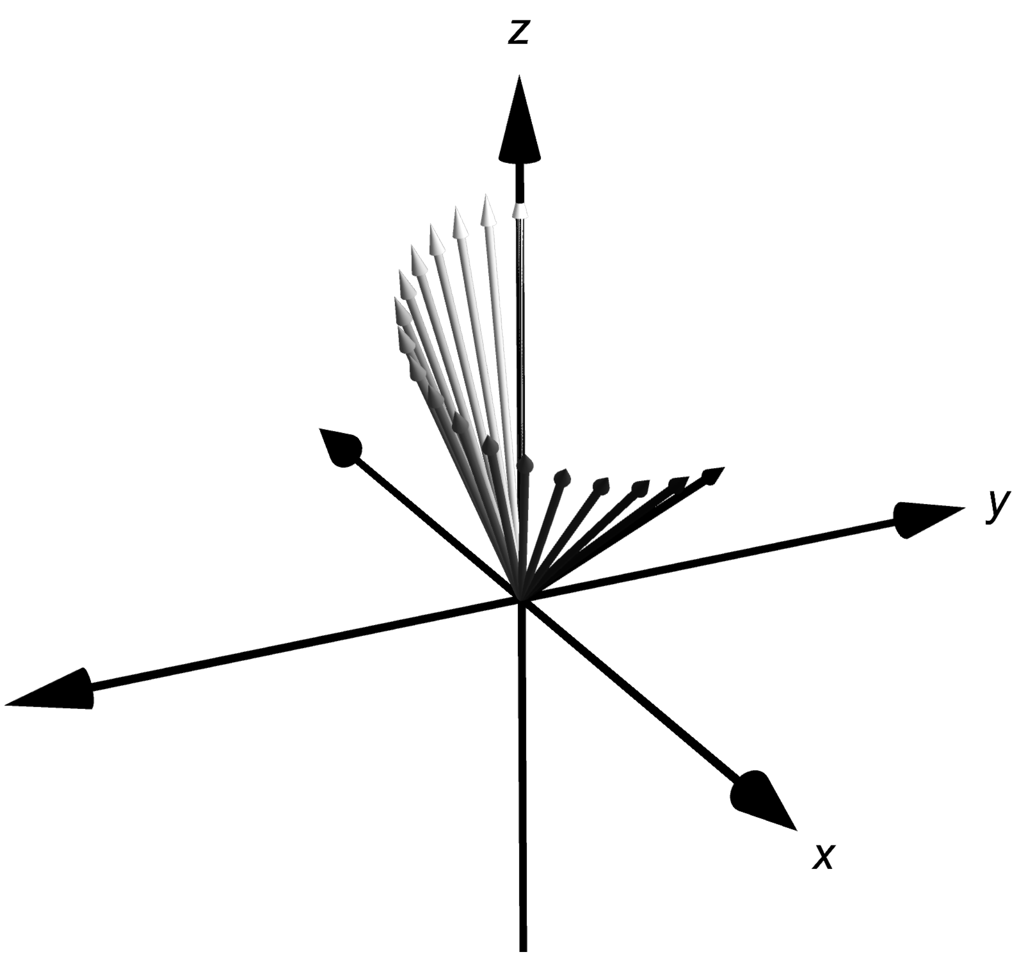
In[22]:=
```
Show[GraphicsGrid @@
  {{my$drawing[ρ$vector[t, #, 1, 0], π / 2, 18]} &
   /@ {0., 1., 2., 3.}}, ImageSize → Full]
```

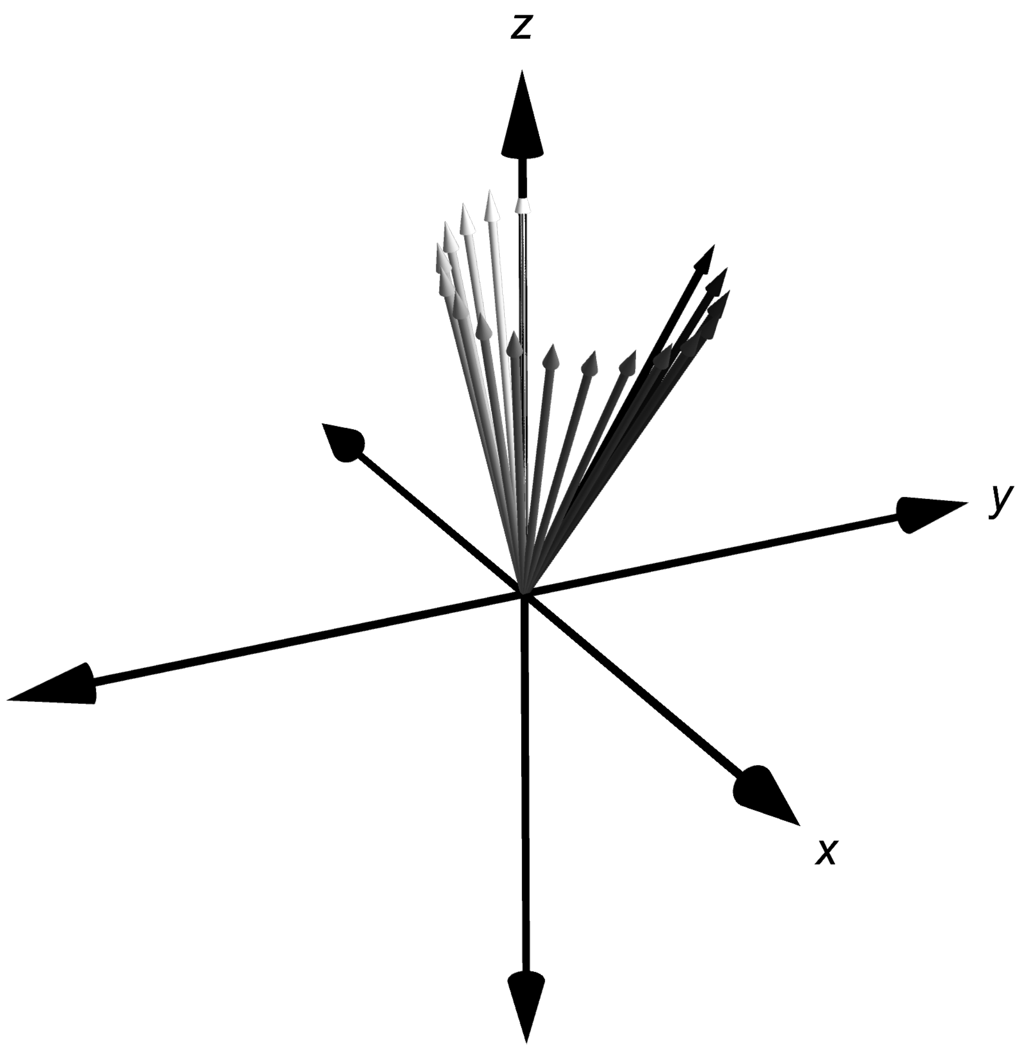

z
y
x


z



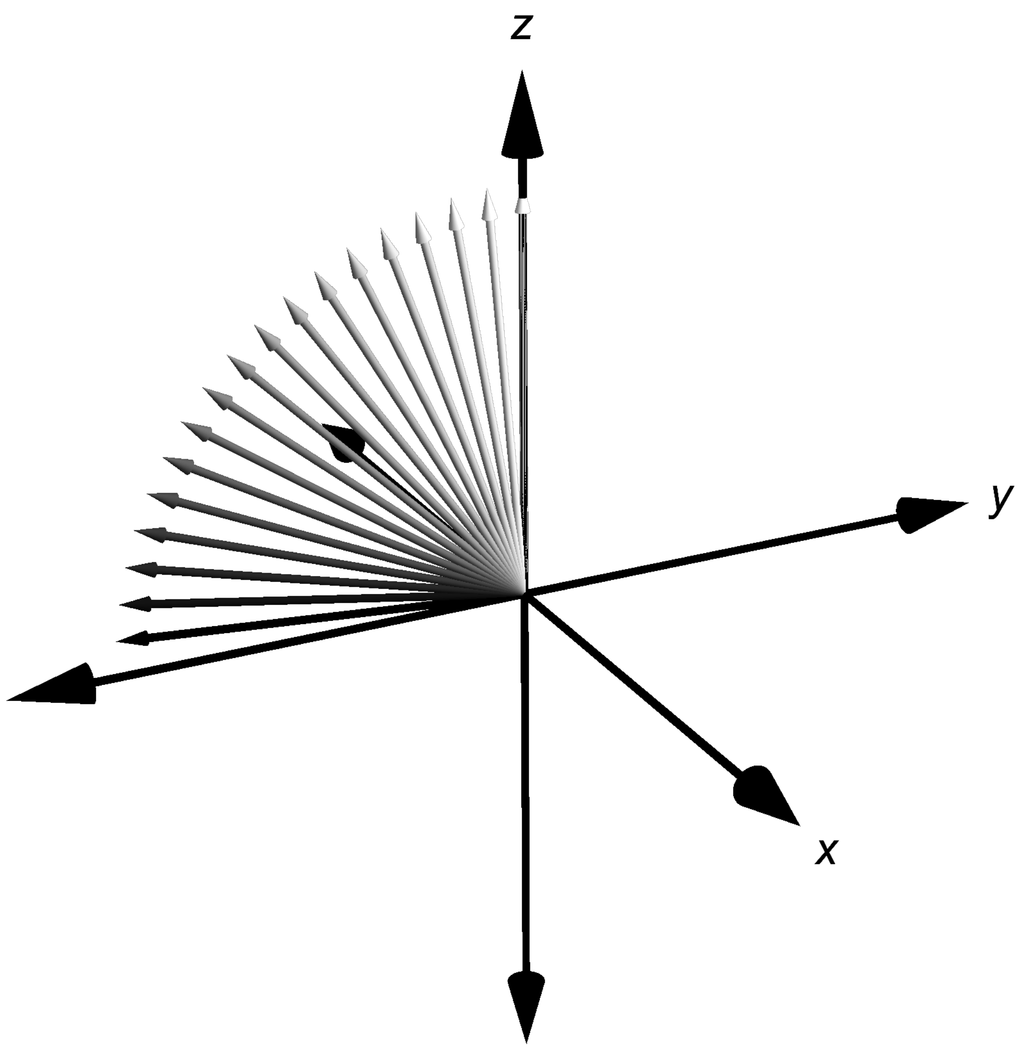

Out[22]=

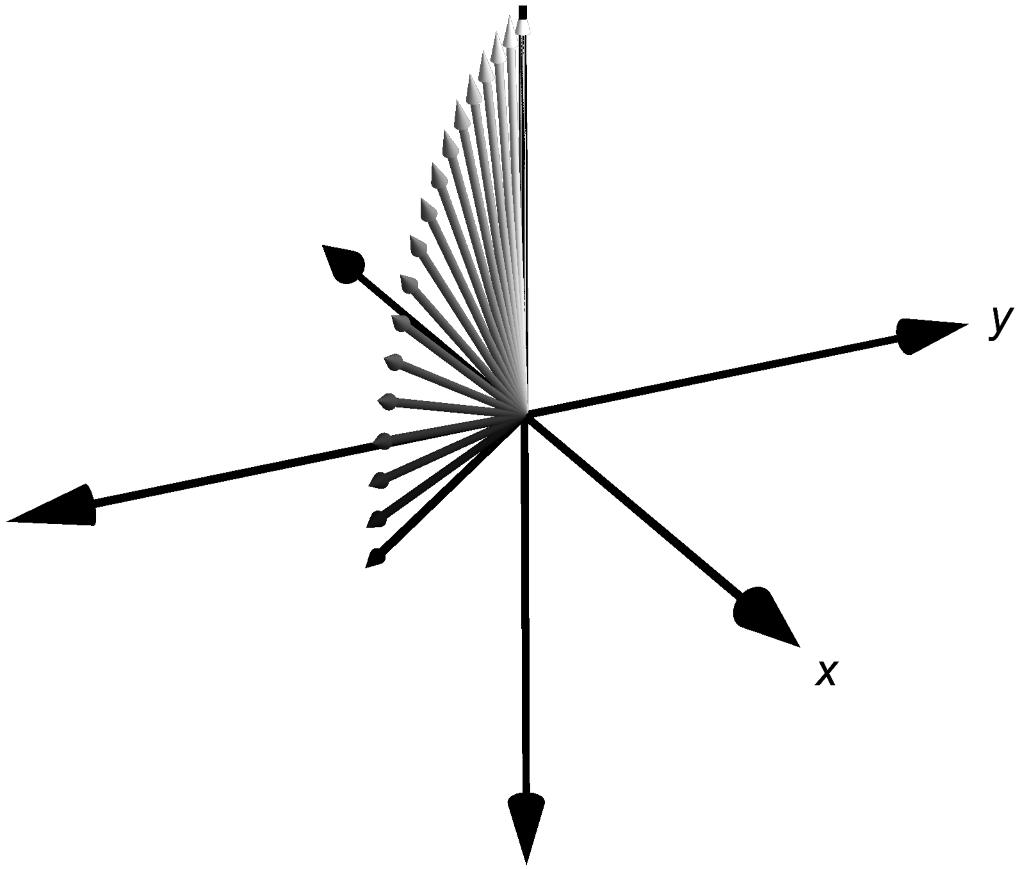




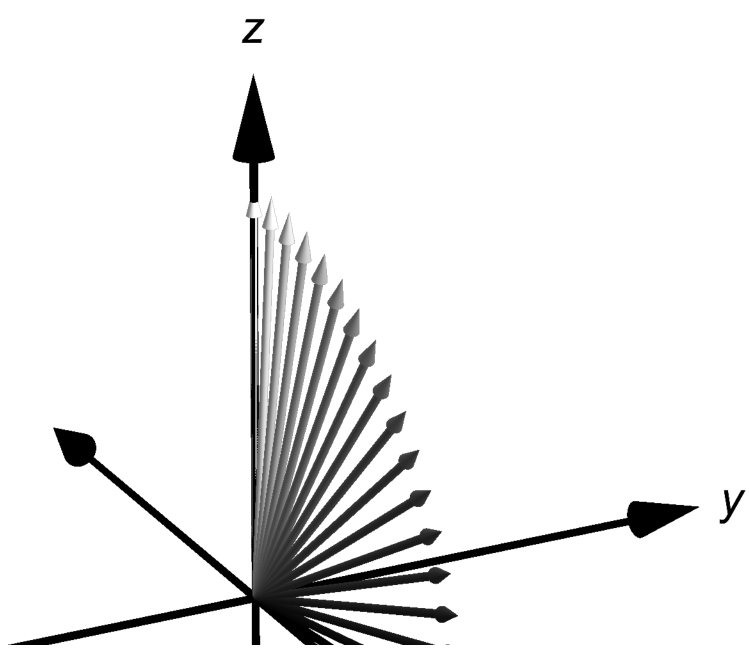

Set the resonance offset d $\Delta = 0$, set the Rabi frequency to $\omega = 1$, and look at magnetization out to times equal to $\pi/2$. Vary the rf phase and plot the magnetization.

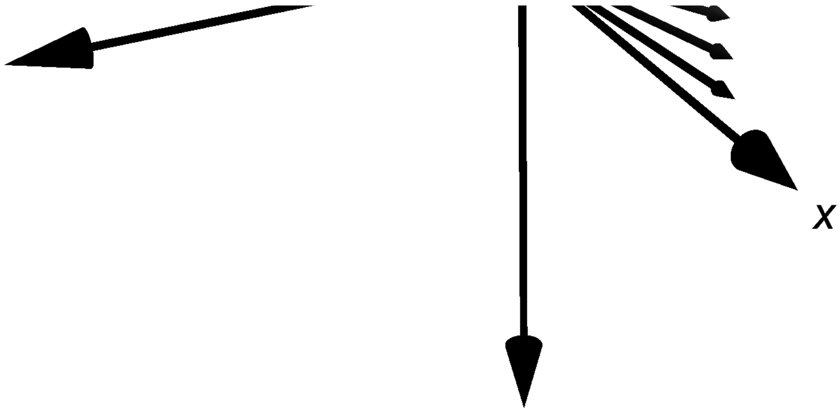
In[23]:=
```
Show[GraphicsGrid @@
  {{my$drawing[ρ$vector[t, 0, 1, #], π/2, 18]} &
   /@ ({0., 45, 90, 180, 270} π/180)}, ImageSize → Full]
```

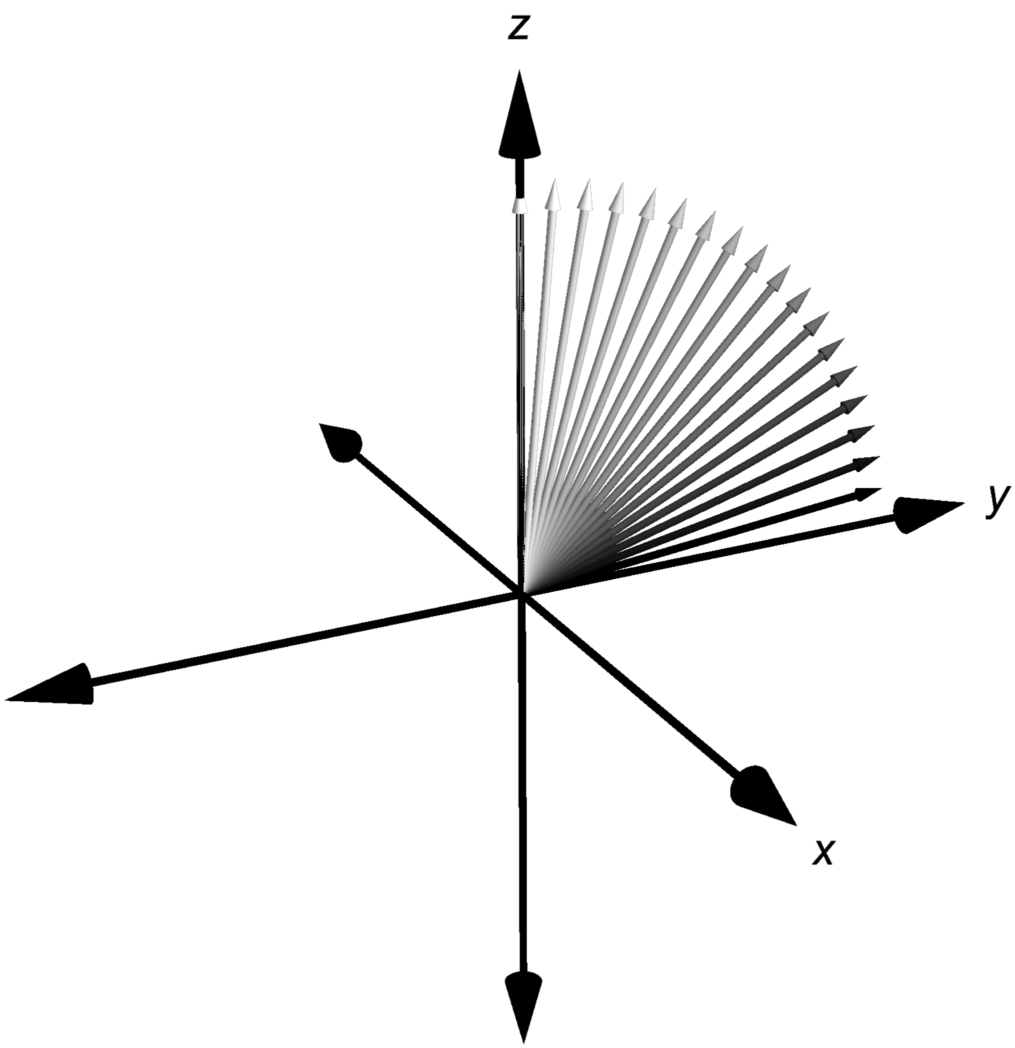


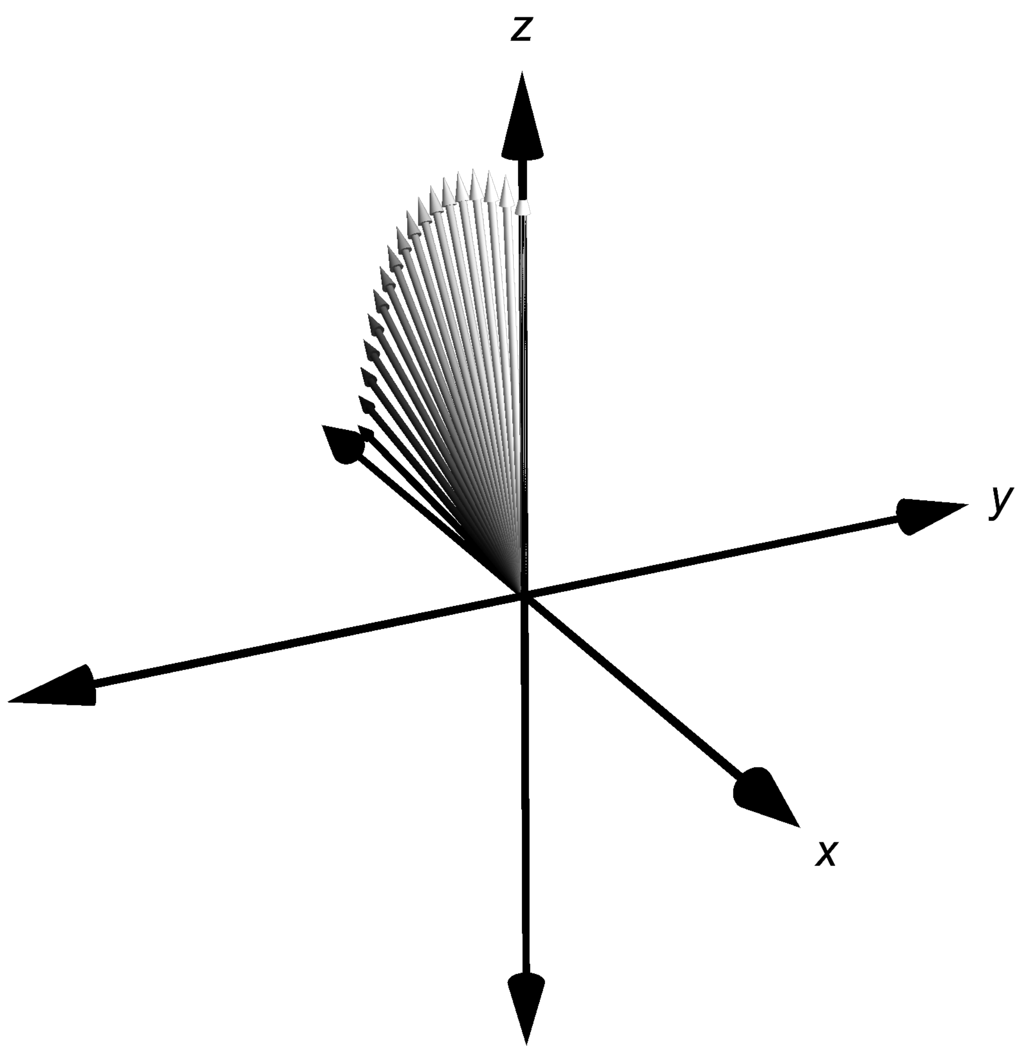




Out[23]=




x


z
y
x




# Clean up

In[24]:=
```
(*
Clear[ω, Δ, ϕ, t, Ix, Iy, Iz, ρ, ρ$0, H]
*)
```