tested 190817 using *SpinDynamica* 3.0.1 under *Mathematica* 11.0

Composite Pulse

this example notebook shows how trajectories of a magnetization vector may be generated under a sequence of events

```
Needs["SpinDynamica`"]
```

```
SpinDynamica version 3.0.1 loaded
```

ModifyBuiltIn: The following built–in routines have been modified in SpinDynamica:
{Chop, Dot, Duration, Exp, Expand, ExpandAll, NumericQ, Plus, Power, Simplify, Times, WignerD}.
Evaluate ??*symbol* to generate the additional definitions for *symbol*.

Trajectory of rotating-frame Magnetization Vectors under a composite pulse, without relaxation

Set up

```
SetSpinSystem[1]
```

SetSpinSystem: the spin system has been set to $\{\{1, \frac{1}{2}\}\}$

SetBasis: the state basis has been set to ZeemanBasis$[\{\{1, \frac{1}{2}\}\}$, BasisLabels → Automatic$]$.

set nutation frequency and pulse durations

```
ωnut = 2 π 50 × 10^3
```

```
100000 π
```

```
τ360 = 2 π / ωnut // N
```

```
0.00002
```

```
τ90 = τ360 / 4;
τ180 = τ360 / 2;
```

set a factor to take into account an rf field error

```
rffactor = 0.9
```

```
0.9
```

define the CompositePulse events

the events in this example have the form {Hamiltonian, duration}

```
CompositePulse = {{rffactor ωnut opI["y"], τ90},
  {rffactor ωnut opI["x"], τ180}, {rffactor ωnut opI["y"], τ90}}
```

$\{\{282743.\ I_{1y}, 5.\times 10^{-6}\}, \{282743.\ I_{1x}, 0.00001\}, \{282743.\ I_{1y}, 5.\times 10^{-6}\}\}$

```
TotalDuration = EventDuration[CompositePulse]
```

```
0.00002
```

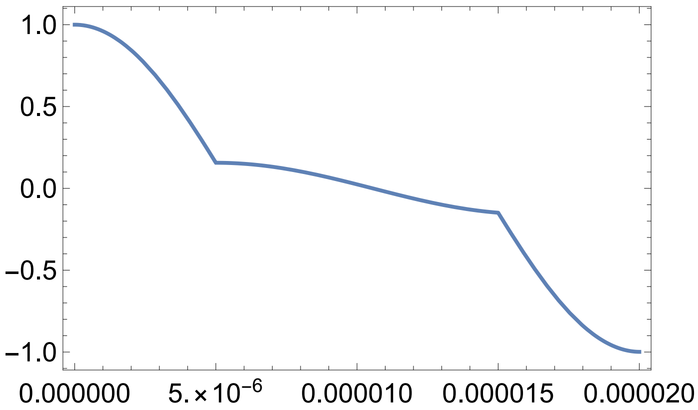
on-resonance trajectory of z-magnetization, starting from z

in this example, an option is given setting the initial time point to 0. Otherwise the final time point will be 0

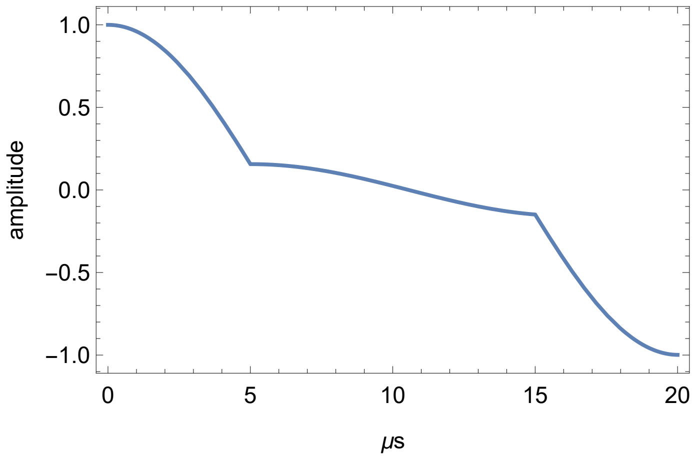
```
Iztraj = Trajectory[
  opI["z"] → opI["z"],
  CompositePulse
 ]
```

```
TrajectoryFunction[ {{0, 20. × 10^-6}} , <>]
```

```
Plot[Iztraj[t], {t, 0, TotalDuration}, Frame → True, PlotStyle → Thick]
```




```
Plot[Iztraj[tμs × 10^-6], {tμs, 0, TotalDuration × 10^6}, Frame → True,
 PlotStyle → Thick, LabelStyle → Directive[Medium, FontFamily → "Helvetica"],
 FrameLabel → {"μs", "amplitude"}]
```

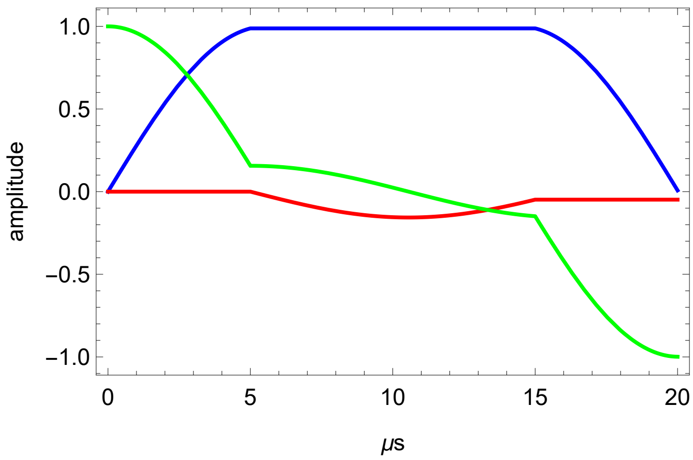


on-resonance trajectory of {x,y,z}-magnetization, starting from z

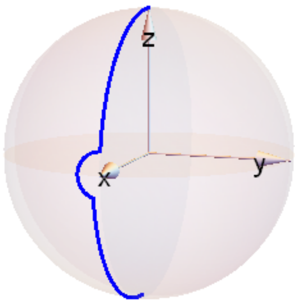
```
{Ixtraj, Iytraj, Iztraj} =
 Trajectory[
  opI["z"] → {opI["x"], opI["y"], opI["z"]},
  CompositePulse
 ]
```

```
{TrajectoryFunction[ {{0, 20. × 10^-6}} , <>],
 TrajectoryFunction[ {{0, 20. × 10^-6}} , <>], TrajectoryFunction[ {{0, 20. × 10^-6}} , <>]}
```

```
Plot[{Ixtraj[tμs × 10^-6], Iytraj[tμs × 10^-6], Iztraj[tμs × 10^-6]},
 {tμs, 0, TotalDuration × 10^6}, Frame → True,
 PlotStyle → {{Thick, Blue}, {Thick, Red}, {Thick, Green}},
 LabelStyle → Directive[Medium, FontFamily → "Helvetica"],
 FrameLabel → {"μs", "amplitude"}
]
```



```
Show[
 Graphics3D[
  {Opacity[0.1], EdgeForm[],
   Polygon[{0, Cos[#], Sin[#]} & /@ Range[0, 2 π, 2 π / 100]],
   Polygon[{Cos[#], Sin[#], 0} & /@ Range[0, 2 π, 2 π / 100]],
   Polygon[{Cos[#], 0, Sin[#]} & /@ Range[0, 2 π, 2 π / 100]],
   Sphere[{0, 0, 0}, 1]
  }
 ],
 ParametricPlot3D[
  Re@Through[{Ixtraj, Iytraj, Iztraj}[t]], {t, 0, TotalDuration},
  Boxed → True, Axes → None, PlotStyle → {{Thick, Blue}}
 ],
 Axes3D[], Boxed → False, ViewPoint → {6, 2, 1}, ViewVertical → ez
]
```

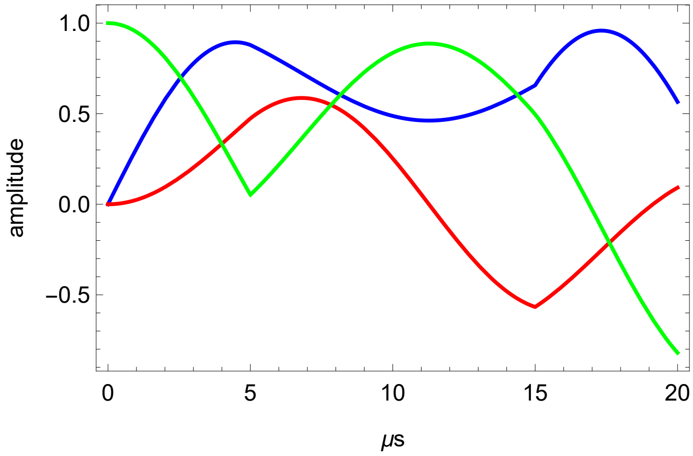


off-resonance trajectory of {x,y,z}-magnetization, starting from z

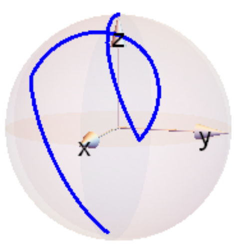
This example uses the option BackgroundGenerator to provide a Hamiltonian term that acts at the same time as the defined events

```
rffactor = 1.0;

resonanceoffset = 0.5 ωnut;

CompositePulse = {{rffactor ωnut opI["y"], τ90},
   {rffactor ωnut opI["x"], τ180}, {rffactor ωnut opI["y"], τ90}};

{Ixtraj, Iytraj, Iztraj} =
 Trajectory[
  opI["z"] → {opI["x"], opI["y"], opI["z"]},
  CompositePulse,
  BackgroundGenerator → resonanceoffset × opI["z"]
 ]
{TrajectoryFunction[ {{0, 20. × 10^-6}} , <>],
 TrajectoryFunction[ {{0, 20. × 10^-6}} , <>], TrajectoryFunction[ {{0, 20. × 10^-6}} , <>]}
```

```
Plot[{Ixtraj[tμs × 10^-6], Iytraj[tμs × 10^-6], Iztraj[tμs × 10^-6]},
 {tμs, 0, TotalDuration × 10^6},
 Frame → True, PlotStyle → {{Thick, Blue}, {Thick, Red}, {Thick, Green}},
 LabelStyle → Directive[Medium, FontFamily → "Helvetica"],
 FrameLabel → {"μs", "amplitude"}
]
```




```
Show[
 Graphics3D[
  {Opacity[0.1], EdgeForm[],
   Polygon[{0, Cos[#], Sin[#]} & /@ Range[0, 2 π, 2 π / 100]],
   Polygon[{Cos[#], Sin[#], 0} & /@ Range[0, 2 π, 2 π / 100]],
   Polygon[{Cos[#], 0, Sin[#]} & /@ Range[0, 2 π, 2 π / 100]],
   Sphere[{0, 0, 0}, 1]
  }
 ],
 ParametricPlot3D[
  Re@Through[{Ixtraj, Iytraj, Iztraj}[t]], {t, 0, TotalDuration},
  Boxed → True, Axes → None, PlotStyle → {{Thick, Blue}}
 ],
 Axes3D[], Boxed → False, ViewPoint → {6, 2, 1}, ViewVertical → ez
]
```

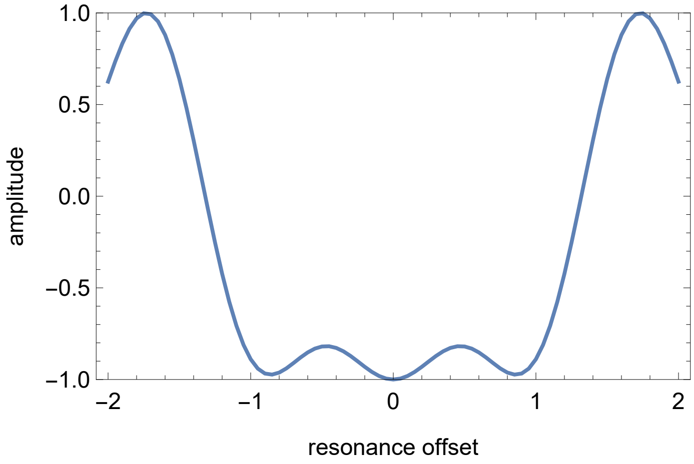


determine off-resonance and rf performance of a composite pulse using TransformationAmplitude

```
rffactor = 1.0;
```

```
resonanceoffset = 0.5 ωnut;
```

```
Clear[CompositePulse];
CompositePulse[rffactor_] :=
 {{rffactor ωnut opI["y"], τ90},
  {rffactor ωnut opI["x"], τ180},
  {rffactor ωnut opI["y"], τ90}
 }

ztoz[rffactor_, resonanceoffset_] :=
 TransformationAmplitude[
  opI["z"] → opI["z"],
  CompositePulse[rffactor],
  BackgroundGenerator → resonanceoffset × ωnut opI["z"]
 ]

ListPlot[
 Table[{off, ztoz[1, off]}, {off, -2, 2, 0.05}],
 Frame → True, PlotRange → {-1, 1}, PlotStyle → Thick,
 Joined → True, LabelStyle → Directive[Medium, FontFamily → "Helvetica"],
 FrameLabel → {"resonance offset", "amplitude"}
]
```



```
ListContourPlot[
 Flatten[Table[{off, rf, ztoz[rf, off]}, {off, -2, 2, 0.1}, {rf, 0, 2, 0.1}], 1],
 FrameLabel → {"resonance offset", "rf field"},
 FrameStyle → Directive[Medium, FontFamily → "Helvetica"]
]
```