A simple spacecraft simulation

This application note describes a pretty simple spacecraft simulation that allows a person to control a spacecraft in empty space by firing rotational and translational thrusters using a joystick with an adapter like that described in the previous analog computer application note (#18).

1 Drawing a spacecraft figure

First of all a spacecraft like shape is required. The basis for most figures to be displayed on an oscilloscope is a unit-circle which can be easily generated by deriving a sine/cosine signal pair by solving the differential equation

$$\ddot{y} + \omega^2 y = 0. \tag{1}$$

Figure 1 shows the corresponding general analog computer setup to solve this equation: The two potentiometers controlling ω may be omitted since just a sine/cosine signal pair with sufficiently high frequency to appear flicker-free on the oscilloscope's screen is required. The potentiometer labeled a may also be omitted by setting the initial condition of the first integrator to $+1$. The two potentiometers a^* and α are crucial as a long running simulation requires a stable amplitude of the sine/cosine signal pair. α provides some positive feedback and should be set to a very small value such as 0.01 or even smaller. a^* limits the amplitude of the output of the first integrator, counteracting the effect of α. These two potentiometers should be set so that a sine/cosine signal pair with stable amplitude of at least 0.5 ($= 5$ V on an Analog Paradigm Model-1 analog computer) is obtained.[1]

Using the $-\cos(10\omega\tau)$ and $-\sin(10\omega\tau)$ outputs of this circuit, a circle can be displayed on an xy-display.[2] The next step is to transform this figure into a shape with a distinguishable "front" and "back" as the spacecraft is, of course, oriented. Using a

[1] This circuit, also called a *quadrature generator*, is so useful that it may be worthwhile to build one as a dedicated computing element thus saving the hassle of setting up this particular circuit over and over again.

[2] τ denotes the machine time which depends on the selected time scale factor of the integrators.

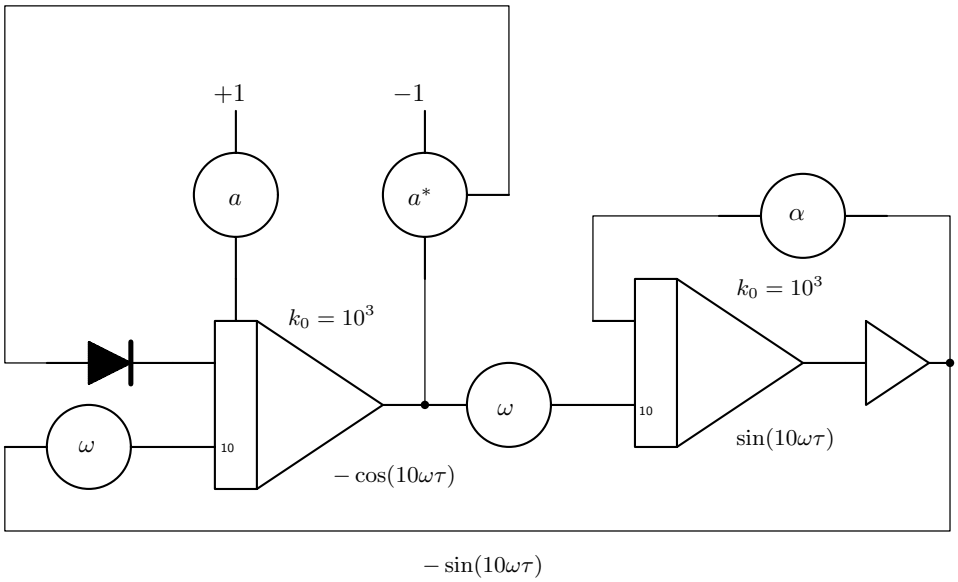

Figure 1: Basic computer setup for equation (1)

function generator would be a suitable approach but it is also possible to perform this operation by standard computing elements as shown in figure 2. The output signals constitute a time-varying shape vector

$$\vec{s} = \begin{pmatrix} s_x \\ s_y \end{pmatrix}$$

the size of which is controlled by the coefficient potentiometers λ_x and λ_y. Suitable values for these are $\lambda_x = 0.04$ and $\lambda_y = 0.05$. Figure 3 shows the resulting shape of the spacecraft.

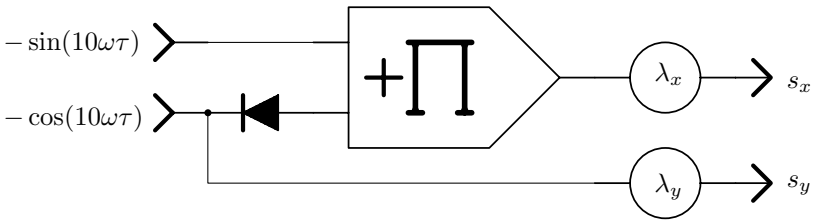

Figure 2: Creating a spacecraft-like shape by deforming a circle

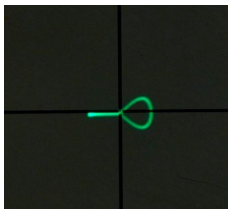

Figure 3: Shape of the spacecraft as displayed on an xy-display

2 Generating $\dot{\varphi}$ and a

Using an analog joystick,[3] the next task is to generate suitable control signals representing the angular velocity $\dot{\varphi}$ and the longitudinal acceleration a as controlled by the joystick. Figure 4 shows the corresponding computer setup with $-1 \leq j_x \leq 1$ and $-1 \leq j_y \leq 1$ denoting the joystick output signals. The two Z-diodes in series have a Zener-voltage of 10 V and limit the output signal of the integrator to avoid an overload condition.

[3] Using comparators and electronic switches in conjunction with one integrator each, even a cheap digital "retro gaming" joystick can be used.

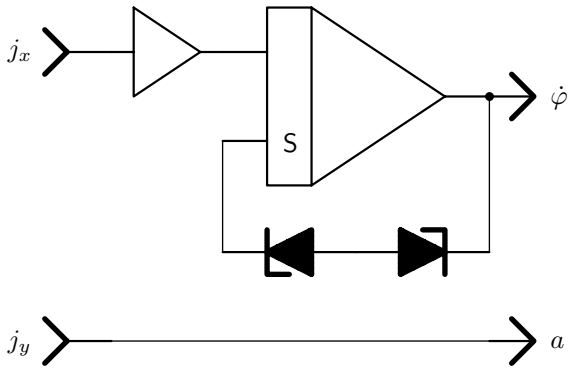

Figure 4: Deriving $\dot{\varphi}$ and a from the joystick interface

3 Rotating the spacecraft

Using the angular velocity $\dot{\varphi}$ it is now possible to derive the corresponding values for $\pm\sin(\phi)$ and $\pm\cos(\phi)$ which will be used to implement a rotation matrix. Figure 5 shows the computer setup yielding these four harmonic functions based on $\dot{\varphi}$.

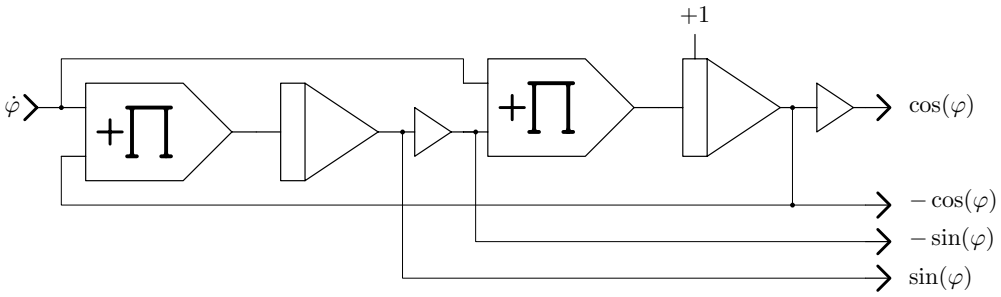

Figure 5: Generating $\pm\sin(\phi)$ and $\pm\cos(\phi)$

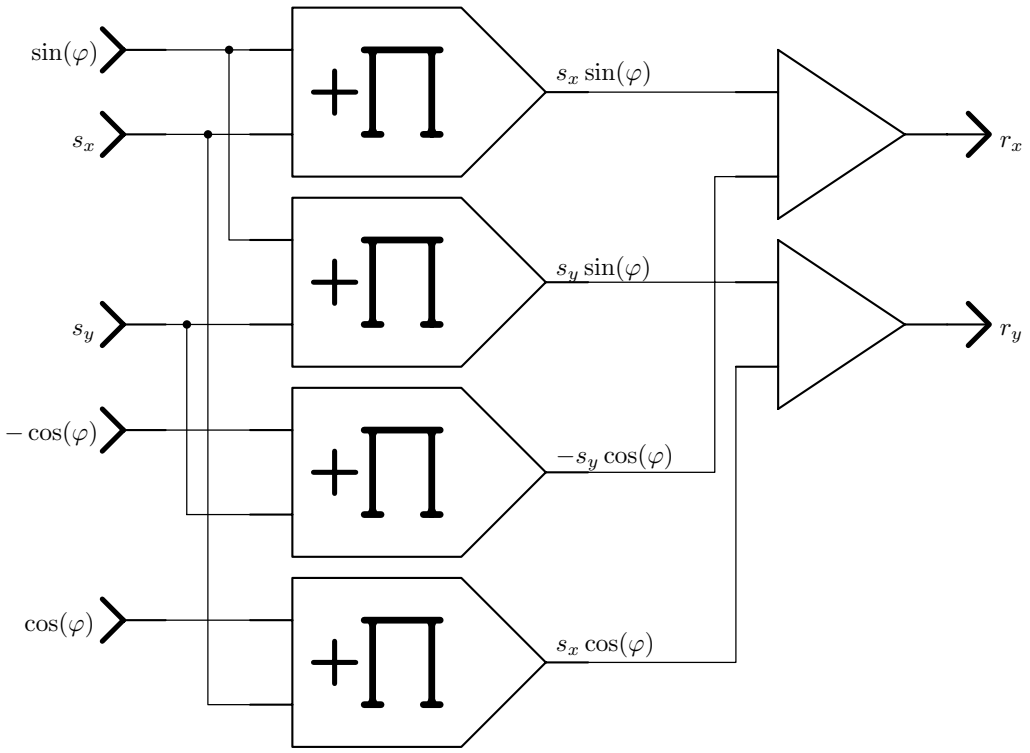

Figure 6: Rotating the spacecraft shape

The actual rotation is then performed by computing

$$\vec{r} = \begin{pmatrix} -\sin(\varphi) & \cos(\varphi) \\ -\cos(\varphi) & -\sin(\varphi) \end{pmatrix} \vec{s}$$

as shown in figure 6, where $\vec{r}$ denotes the time-varying vector of the rotated spacecraft shape.

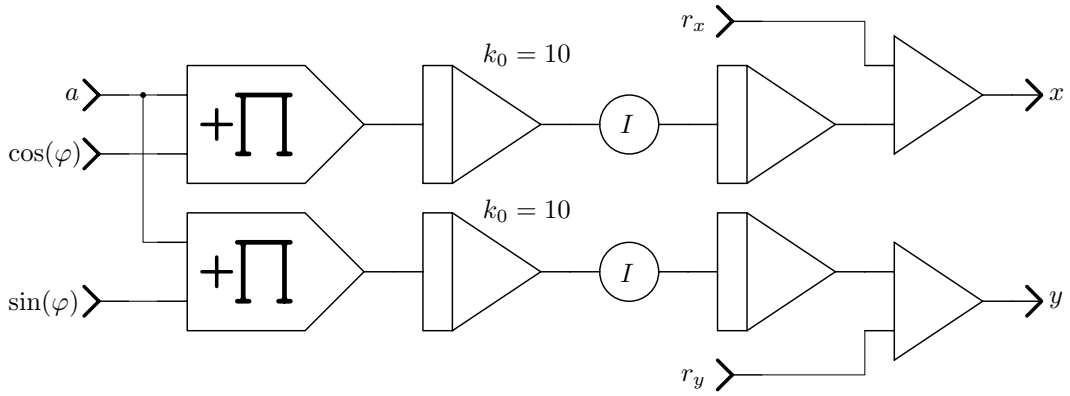

Figure 7: Moving the spacecraft around

4 Moving the spacecraft around

This rotated spacecraft shape must now be able to move around on the screen of an xy-display. This is accomplished by "firing" the longitudinal rocket engines (front or back of the spacecraft, depending on the movement of the joystick). Therefore, the acceleration due to these rocket engines must be integrated twice in a component-wise fashion according to the current angle of attack φ of the spacecraft as shown in figure 7. The two potentiometers labeled I control the inertia of the spacecraft – useful values (depending on the skill of the "pilot") are $0.05 \leq I \leq 0.15$.

5 The overall implementation

The overall implementation of the spacecraft simulation is shown in figure 8. All in all, the following complement of computing elements is required:

- Eight summers,
- nine integrators,
- six coefficient potentiometers,

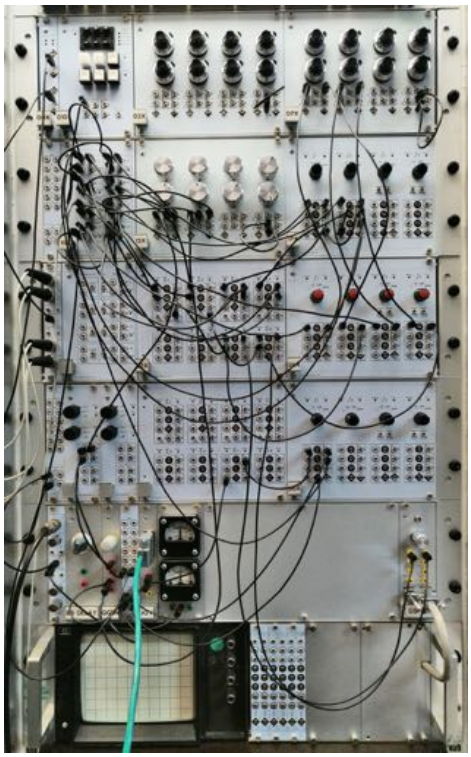

Figure 8: Overall setup of the simple spacecraft simulation

- nine multipliers,
- two diodes,
- two 10 V Z-diodes,
- one xy-display,
- and a joystick controller like that described in our analog computer application note #18.

Happy analog computing!