```
function [t_out, x_out, P_out, v_out, Pvv_out] = sris(F, Gamma, H, Q, R,
xhat0, P0, z)

%SRIS Square-root Information Smoother

% Will need to save matrices on the forward pass

% Characterize the data
nx = length(xhat0);
nz = size(z, 2);
nv = size(Q, 1);
num_meas = size(z, 1)/size(H, 1);

% Initialize forward pass outputs
t_out = [0:num_meas].';
x_forward = zeros(num_meas*2 + 1, length(xhat0));
x_forward(1, :) = xhat0;
P_forward = zeros([size(P0), num_meas*2 + 1]);
P_forward(:, :, 1) = P0;

% Initialize saving arrays for smoothing operations
Rvv_bar_forward = zeros(nv, nv, num_meas);
Rvx_bar_forward = zeros(nv, nx, num_meas);
Rxx_bar_forward = zeros(nx, nx, num_meas);
zx_forward = zeros(num_meas, nx);
zv_bar_forward = zeros(num_meas, nv);

% Initialize the smoother outputs
x_out = zeros(num_meas + 1, nx);
P_out = zeros(nx, nx, num_meas + 1);
v_out = zeros(num_meas+1, nv);
Pvv_out = zeros(nv, nv, num_meas + 1);

% Compute some constants
Ra = chol(R); % This is not necessarily the same for every measurement in
general
Rait = inv(Ra).';
G = zeros(nx, 1); % No inputs for now
u = 0; % No inputs for now

% Set up initial priors
Rxx = inv(chol(P0)).'; % Smarter way to do this in the notes
Rvv = inv(chol(Q)).';
vhat = zeros(size(Q, 1), 1);
zv = Rvv*vhat;
zx = Rxx*xhat0;
```

Run the SRIF Forward

```
for k=1:num_meas

    % Propagate
    lower_left = -Rxx*(F\Gamma);
    lower_right = Rxx/F;
    [Qb, Rb] = qr([Rvv, zeros(nv, nx); lower_left, lower_right]);

    bottom = zx + Rxx*(F\G)*u;
    zb = Qb.'*[zeros(nv, 1); bottom];

    % Extract block elements from matrices
    zv_bar_forward(k, :) = zb(1:nv).'; % Save for smoothing
    zx_bar = zb(end-nx+1:end); % I hate matlab indexing
    Rxx_bar = Rb(end-nx+1:end, end-nx+1:end);
    Rxx_bar_forward(:, :, k) = Rxx_bar;
    Rvv_bar_forward(:, :, k) = Rb(1:nv, 1:nv); % Save for smoothing
    Rvx_bar_forward(:, :, k) = Rb(1:nv ,end-nx+1:end);

    % Save P_bar and x_bar
    x_forward(2*k, :) = Rxx_bar\zx_bar;
    P_forward(:, :, 2*k) = inv(Rxx_bar)*inv(Rxx_bar).';

    % Update
    zk = z(k, :).';
    za = Rait*zk;
    Ha = Rait*H;

    [Qc, Rc] = qr([Rxx_bar; Ha]);

    zc = Qc.'*[zx_bar; za];

    % Extract block elements from matrices
    Rxx = Rc(1:nx, 1:nx);
    zx = zc(1:nx);
    zx_forward(k, :) = zx; % Save for smoothing

    % Save P and x_hat
    x_forward(2*k + 1, :) = Rxx\zx;
    P_forward(:, :, 2*k + 1) = inv(Rxx)*inv(Rxx).';

end
```

Run the smoother backwards

Save the final forward estimates

```
x_out(end, :) = x_forward(end, :);
P_out(:, :, end) = P_forward(:, :, end);

% Start with the final values
% Rxx carries over from srif
Rvv_bar = Rvv_bar_forward(:, :, end);
Rvx_bar = Rvx_bar_forward(:, :, end);
zx = zx_forward(end, :).';
zv = zv_bar_forward(end, :).';
```

```
% Iterate
for i=num_meas:-1:1

    % Build the block matrix
    upper_left = Rvv_bar + Rvx_bar*Gamma;
    upper_right = Rvx_bar*F;
    lower_left = Rxx*Gamma;
    lower_right = Rxx*F;
    block_smooth = [upper_left, upper_right; lower_left, lower_right];

    % Build the z vector
    z_smooth = [zv - Rvx_bar*G*u; zx - Rxx*G*u];

    % Perform the QR Factorization to isolate x
    [Q_smooth, R_smooth] = qr(block_smooth);
    z_smooth_post = Q_smooth.'*z_smooth;

    % Extract the relevant values
    Rvv_star = R_smooth(1:nv, 1:nv);
    Rvx_star = R_smooth(1:nv, end-nx+1:end);
    Rxx_star = R_smooth(end-nx+1:end, end-nx+1:end);
    zv_star = z_smooth_post(1:nv);
    zx_star = z_smooth_post(end-nx+1:end);

    % Calculate the smoothed estimates
    x_star = Rxx_star\zx_star;
    v_star = Rvv_star\(zv_star - Rvx_star*x_star);
    P_star = Rxx_star\inv(Rxx_star).';
    % Pvv_star =

    % Save smoothed estimates
    x_out(i, :) = x_star;
    P_out(:, :, i) = P_star;
    v_out(i, :) = v_star;
    % Pvv_out(:, :, i)

    % Update for next iteration
    if i > 1  % Nothing to reassign on last iteration
        Rxx = Rxx_star; % From last smoothing iteration
        Rvv_bar = Rvv_bar_forward(:, :, i-1); % Use from forward pass
        Rvx_bar = Rvx_bar_forward(:, :, i-1); % Use from forward pass
        zx = zx_star; % Use from last smoothing iteration
        zv = zv_bar_forward(i-1, :).'; % Use from forward pass
    end

end

end
```

Published with MATLAB® R2025a