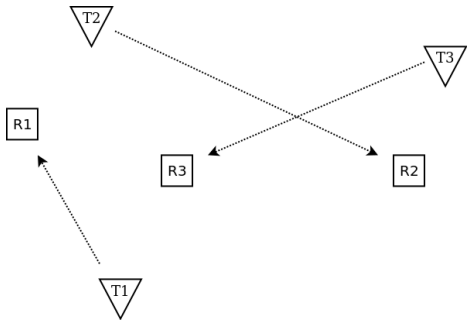
From R to Julia: Converting Workshop Code

Mick Cooney
michael.cooney@applied.ai

10 March 2016

Power Control Algorithm



- Network of n transmitter/receiver pairs
- Power level: $p_i > 0$, Gain: $G_{ij} > 0$, Threshold: γ
- Signal power at receiver i: $s_i = G_{ii} p_i$.
- Noise plus interference: $q_i = \sigma + \sum_{j \neq i} G_{ij} p_j$
- SINR: $S_i = \frac{s_i}{q_i} = \alpha\gamma$, safety margin: α

Simple power update algorithm:

$$p_i(t+1) = p_i(t)\left(\frac{\alpha\gamma}{S_i(t)}\right)$$

Rearrange in matrix form:

$$\begin{bmatrix} p_1(t+1) \\ p_2(t+1) \\ p_3(t+1) \end{bmatrix} = \begin{bmatrix} 0 & \frac{\alpha\gamma G_{12}}{G_{11}} & \frac{\alpha\gamma G_{13}}{G_{11}} \\ \frac{\alpha\gamma G_{21}}{G_{22}} & 0 & \frac{\alpha\gamma G_{23}}{G_{22}} \\ \frac{\alpha\gamma G_{31}}{G_{33}} & \frac{\alpha\gamma G_{32}}{G_{33}} & 0 \end{bmatrix} \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \end{bmatrix} + \begin{bmatrix} \frac{\alpha\gamma\sigma}{G_{11}} \\ \frac{\alpha\gamma\sigma}{G_{22}} \\ \frac{\alpha\gamma\sigma}{G_{33}} \end{bmatrix}$$

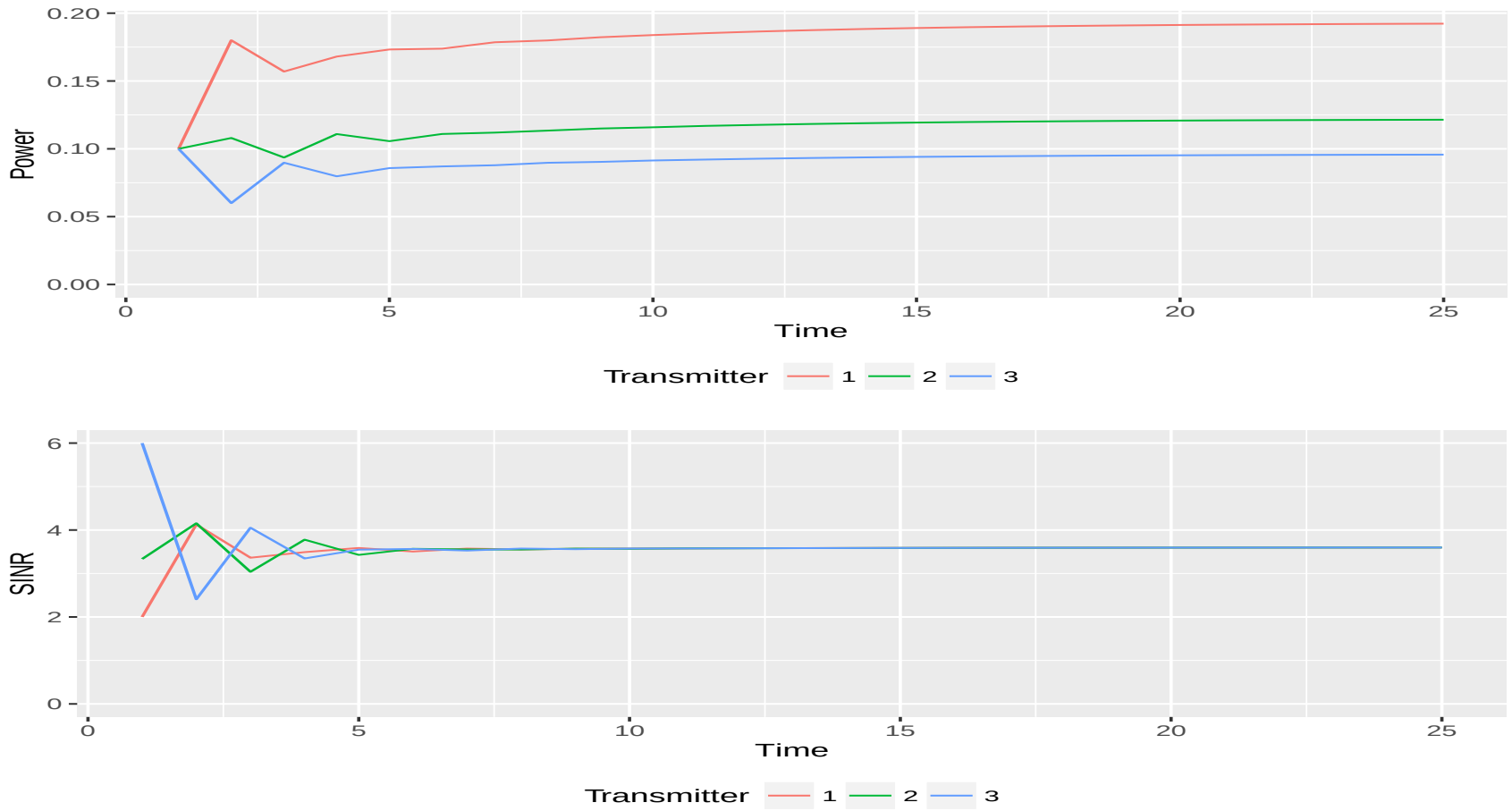
$$p_i(t+1) = Ap(t) + b$$

Plot SINR and power output over time

```
G <- matrix(c(1.0, 0.2, 0.2,
              0.1, 2.0, 0.4,
              0.3, 0.1, 3.0), ncol = 3, byrow = TRUE);

gamma <- 3.0;
alpha <- 1.2;
sigma <- 0.01;

N <- dim(G)[1];

mask  <- 1 - diag(N);
numer <- alpha * gamma * G;
denom <- matrix(rep(diag(G), N), ncol = N);

A <- mask * (numer / denom)

b <- alpha * gamma * sigma / diag(G)

q_mat <- mask * G;

n_iter <- 25;

pout    <- matrix(0, ncol = n_iter, nrow = N);
SINRout <- matrix(0, ncol = n_iter, nrow = N);

p0 <- rep(0.1, N);

pout[,1]    <- p0;
q           <- sigma + q_mat %*% p0;
SINRout[,1] <- (diag(G) * pout[,1]) / q;
```

```
for(i in 1:(n_iter-1)) {
    pout[,i+1] <- A %*% pout[,i] + b;

    q <- sigma + q_mat %*% pout[,i+1];

    SINRout[,i+1] <- (diag(G) * pout[,i+1]) / q;
}

power.plot <- qplot(Var2, value, data = melt(pout), geom = 'line', colour = as.character(Var1), size = I(0.5)) +
    xlab('Time') + ylab('Power') +
    expand_limits(y = 0) +
    theme(legend.position = 'bottom') +
    scale_colour_discrete(name = 'Transmitter');

sinr.plot <- qplot(Var2, value, data = melt(SINRout), geom = 'line', colour = as.character(Var1), size = I(0.5)) +
    xlab('Time') + ylab('SINR') +
    expand_limits(y = 0) +
    theme(legend.position = 'bottom') +
    scale_colour_discrete(name = 'Transmitter');
```

```
grid.arrange(power.plot, sinr.plot, nrow = 2);
```



Julia Code

```
G = [1.0 0.2 0.2; 0.1 2.0 0.4; 0.3 0.1 3.0];

N = size(G)[1];
K = 25; # Number of iterations of the circuit

gamma = 3.0;
alpha = 1.2;
sigma = 0.01;

A = ((alpha * gamma * G) .* (ones(3,3) - eye(3))) ./ repmat(diag(G), 1, 3);

b = alpha * gamma * diag(G);

p    = zeros(N, K);
SINR = zeros(N, K);


p[:,1]    = [0.1 0.1 0.1];
q         = sigma + (G - diagm(diag(G))) * p[:,1];
SINR[:,1] = diag(G)' * p[:,1] ./ q;

for i = 2:K
    p[:,i]    = A * p[:,i-1] + b;
    q         = sigma + (G - diagm(diag(G))) * p[:,i];
    SINR[:,i] = diag(G)' * p[:,i] ./ q;
end
```

Summary

michael.cooney@applied.ai

Slides and code available on BitBucket:
`https://www.bitbucket.org/kaybenleroll/dublin_r_workshops`