From R to Julia: Converting Workshop Code

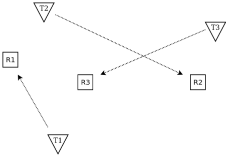
Mick Cooney
michael.cooney@applied.ai

10 March 2016

Background

- Co-organiser of Dublin R
- Give regular workshops on various topics
- Linear Dynamical Systems / Gaussian Processes
- Heavy linear algebra, ideal for translation

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:

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

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

R Code

```
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;
```

R Code

```
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')
```

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 = 50; # 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 * sigma ./ 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
```

Temperatures in a Multicore Processor

Temperature of a process at two locations $T = (T_1, T_2)$

Affine functions of the power dissipated by three cores denoted $P = (P_1, P_2, P_3)$

P_1	P_2	P_3	T_1	T_2
10W	10W	10W	27C	29C
100W	10W	10W	45C	37C
10W	100W	10W	41C	49C
10W	10W	100W	35C	55C

R Code

```r
C <- matrix(c( 10,  10,  10,   0,   0,   0, 1, 0
             ,  0,   0,   0,  10,  10,  10, 0, 1
             ,100,  10,  10,   0,   0,   0, 1, 0
             ,  0,   0,   0, 100,  10,  10, 0, 1
             , 10, 100,  10,   0,   0,   0, 1, 0
             ,  0,   0,   0,  10, 100,  10, 0, 1
             , 10,  10, 100,   0,   0,   0, 1, 0
             ,  0,   0,   0,  10,  10, 100, 0, 1),
            byrow = TRUE, ncol = 8, nrow = 8)

d <- c(27, 29, 45, 37, 41, 49, 35, 55)

output <- solve(C, d)

A <- matrix(output[1:6], byrow = TRUE, ncol = 3)
b <- output[7:8]
```

Julia Code

```
C = [ 10  10  10   0   0   0 1 0;
       0   0   0  10  10  10 0 1;
     100  10  10   0   0   0 1 0;
       0   0   0 100  10  10 0 1;
      10 100  10   0   0   0 1 0;
       0   0   0  10 100  10 0 1;
      10  10 100   0   0   0 1 0;
       0   0   0  10  10 100 0 1]


d = [27; 29; 45; 37; 41; 49; 35; 55]

output = C \ d

A = [output[1:3]'; output[4:6]']
b = output[7:8]

(70 - b) ./ mapslices(sum, A, 2)
```

Concentration of Chemicals in Reaction Kinetics

Reaction chain:

$$C_1 \xrightarrow{k_1} C_2 \xrightarrow{k_2} C_3$$

Model the mixture proportions as a linear system:

$$\dot{x} = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix} x$$

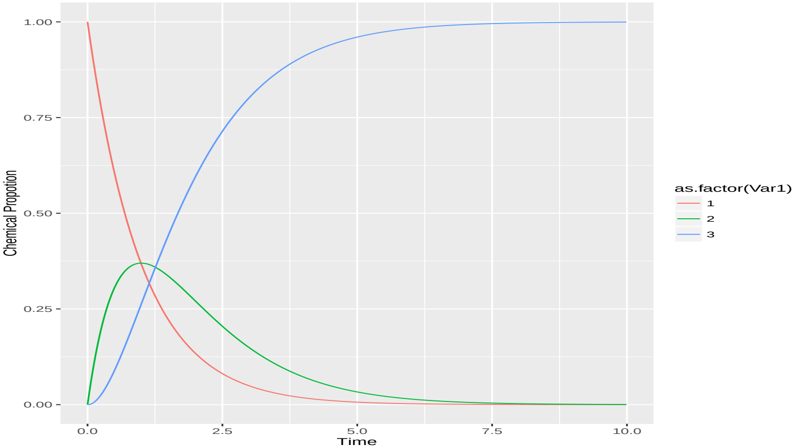
Use timestep h small to get:

$$x(t+1) = (I + hA)\, x(t)$$

R Code

```
k1 <- 1
k2 <- 1

A <- matrix(c(-k1, k1, 0, 0, -k2, k2, 0, 0, 0), ncol = 3)
h <- 0.01

A_update <- (diag(3) + h * A)
n_steps  <- 1000

x <- matrix(0, ncol = n_steps, nrow = 3)

x[, 1] <- c(1, 0, 0)


for(i in 2:n_steps) {
    x[, i] <- A_update %*% x[, i-1]
}
```

```
qplot((Var2 - 1) * h, value, data = melt(x), geom = 'line', colour = as.factor(Var1)
      ,xlab = 'Time'
      ,ylab = 'Chemical Propotion')
```



Julia Code

```
k1 = 1
k2 = 2

h = 0.01

A = [-k1   0 0;
      k1 -k2 0;
       0  k2 0]

A_update = eye(3) + h * A
n_steps  = 1000

x = zeros(3, n_steps)

x[:,1] = [1; 0; 0]

for i = 2:n_steps
    x[:,i] = A_update * x[:,i-1]
end
```

Optimal Control of a Mass Unit

Optimal control problem for a force acting on a unit mass

Unit mass at position $p(t)$, velocity $\dot{p}(t)$, force $f(t)$, where $f(t) = x_i$ for $i1 < t \leq i$, for $i = 1, ..., 10$.

(a) Assume the mass has zero initial position and velocity: $p(0) = \dot{p}(0) = 0$. Minimise $\int_0^{t=10} f(t)^2 \, dt$ subject to: $p(10) = 1$, $\dot{p}(10) = 0$, and $p(5) = 0$.

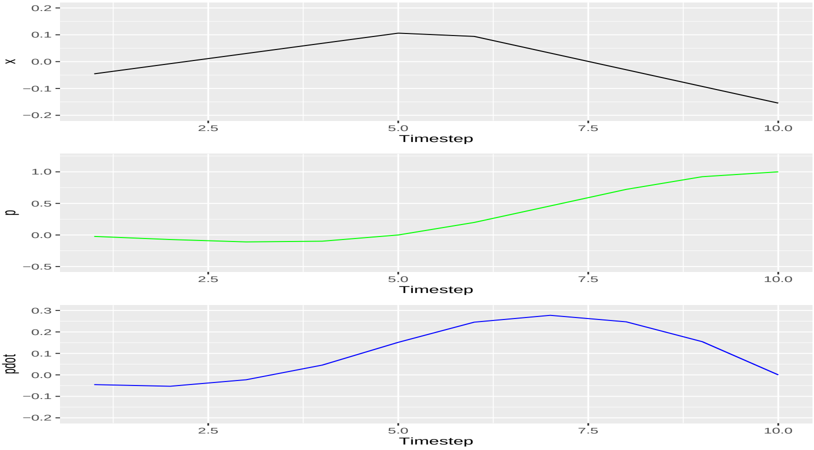
Plot the optimal force f and the resulting p and $\dot{p}$

(b) Assume the mass has initial position $p(0) = 0$ and velocity $\dot{p}(0) = 1$. Our goal is to bring the mass near or to the origin at $t = 10$, at or near rest, i.e. we want $J_1 = p(10)^2 + \dot{p}(10)^2$ small, while keeping $J_2 = \int_0^{t=10} f(t)^2 \, dt$ small, or at least not too large.

Plot the optimal trade-off curve between J_1 and J_2

R Code

```
p10  <- seq(9.5, 0.5, by = -1);
pd10 <- rep(1, 10);
p0   <- c(seq(4.5, 0.5, by = -1), rep(0, 5));

A <- rbind(p10, pd10, p0);
y <- c(1, 0, 0);


x <- MASS::ginv(A) %*% y;
sqrt(sum(x * x))

## [1] 0.249241

x <- corpcor::pseudoinverse(A) %*% y;
sqrt(sum(x * x))

## [1] 0.249241

T1 <- pracma::Toeplitz(rep(1, 10), c(1, rep(0, 9)));
pdot <- T1 %*% x;


T2 <- pracma::Toeplitz(rev(p10), c(0.5, rep(0, 9)));
p <- T2 %*% x;
```


x
0.2
0.1
0.0
−0.1
−0.2
p
1.0
0.5
0.0
−0.5
pdot
0.3
0.2
0.1
0.0
−0.1
−0.2
2.5
5.0
7.5
10.0
Timestep

Gaussian Processes

Summary

michael.cooney@applied.ai

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