

BUGS/WBDiff software: Bayesian inference for dynamical systems

Dave Lunn

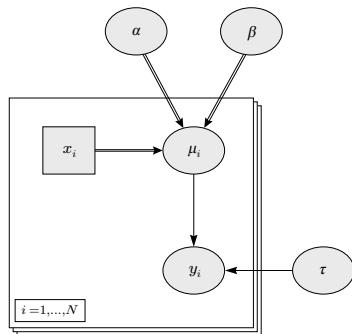
MRC Biostatistics Unit, Cambridge, UK

SBML Workshop: EMBL-EBI, Hinxton, June 20–22, 2011

Outline

- ▶ The BUGS project
 - ▶ mathematical framework – graphical models/Gibbs sampling
 - ▶ underlying philosophy
- ▶ WinBUGS Differential Interface (WBDiff)
 - ▶ illustration with pharmacokinetic models
- ▶ Forced dynamical systems
 - ▶ illustration with artificial pancreas data
- ▶ Conclusions + future work...

Graphical models: linear regression example



$$y_i \sim N(\mu_i, \tau^{-1})$$

$$\mu_i = \alpha + \beta x_i$$

$$i = 1, \dots, N$$

$$\alpha \sim p(\alpha)$$

eg $N(0, 100^2)$

$$\beta \sim p(\beta)$$

eg $N(0, 100^2)$

$$\tau \sim p(\tau)$$

eg $\text{Ga}(\epsilon, \epsilon)$

Why?

- ▶ Can describe (pictorially) very wide class of models with *Directed Acyclic Graphs* (DAGs) – links are *directed* and there are no *cycles*
- ▶ Obvious benefit when models become complicated
- ▶ Convey essential structure of problem without recourse to large set of equations
- ▶ Achieved through abstraction – hiding of detail
- ▶ Graph encodes series of conditional independence assumptions

$$v \perp\!\!\!\perp \text{non-descendants}[v] \mid \text{parents}[v]$$

which allow properties of model to be derived abstractly – more later...

Linear regression example

- Bayes' Theorem:

$$p(\theta|y) \propto p(\theta)p(y|\theta)$$

$$p(\alpha, \beta, \tau|y) \propto p(\alpha)p(\beta)p(\tau) \prod_{i=1}^N N(\alpha + \beta x_i, \tau^{-1})$$

- Full conditional distributions (for Gibbs sampling):

$$p(\alpha|\beta, \tau, y) \propto p(\alpha) \prod_{i=1}^N N(\alpha + \beta x_i, \tau^{-1})$$

$$p(\beta|\alpha, \tau, y) \propto p(\beta) \prod_{i=1}^N N(\alpha + \beta x_i, \tau^{-1})$$

$$p(\tau|\alpha, \beta, y) \propto p(\tau) \prod_{i=1}^N N(\alpha + \beta x_i, \tau^{-1})$$

More generally...

- For *any* DAG:

$$p(V) = \prod_{v \in V} p(v | \text{parents}[v]) \quad [\textit{factorization theorem}]$$

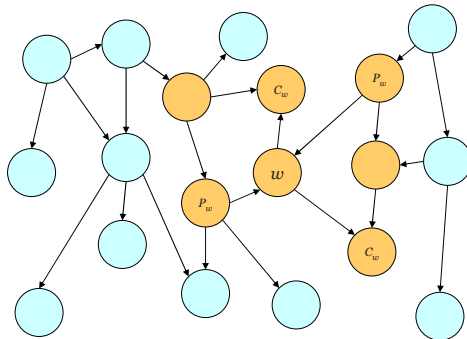
where V is the set of all nodes

- Note that $p(\theta|y) \propto p(\theta, y) = p(V)$
- Also $\text{FCD}(w) = p(w | V \setminus w) \propto p(V)$

$$\Rightarrow \text{FCD}(w) \propto p(w | \text{parents}[w]) \times \prod_{v \in \text{children}[w]} p(v | \text{parents}[v])$$

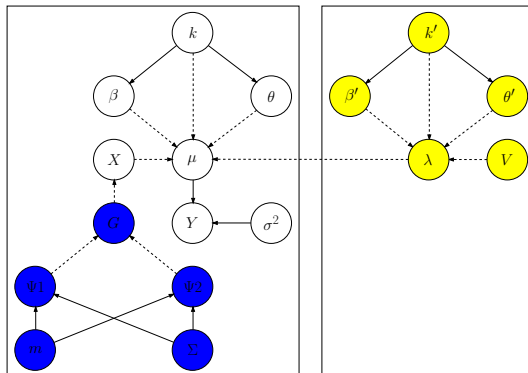
Factorization theorem

- ▶ Beauty of FT is two-fold:
 - (i) can write down joint posterior for any DAG simply by knowing relationship between each node and its parents
 - (ii) full conditional is *local computation* on the graph, involving only the node-parent dependencies for node of interest and its children



Combining models

- ▶ Only need to consider small part of model at any given time; no need to take account of bigger picture...
- ▶ Can construct arbitrarily complex structures by combining submodels together – mechanism of inference remains the same

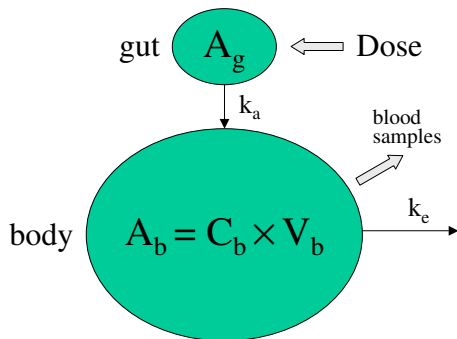


BUGS

- ▶ BUGS: **B**ayesian inference **U**sing **G**ibbs **S**ampling
- ▶ Provides language for specifying parent-child relationships
- ▶ Uses (inverts) these to calculate full conditional distributions

```
model {  
  for (i in 1:N) {  
    y[i] ~ dnorm(mu[i], tau)  
    mu[i] <- alpha + beta * x[i]  
  }  
  alpha ~ dnorm(0, prec)  
  beta ~ dnorm(0, prec)  
  tau ~ dgamma(a, b)  
}  
  
list(  
  N = 20,  
  prec = 0.0001,  
  a = 0.001, b = 0.001,  
  y = c(1.3, 2.4, ...),  
  x = c(9.7, 5.9, ...)  
)
```

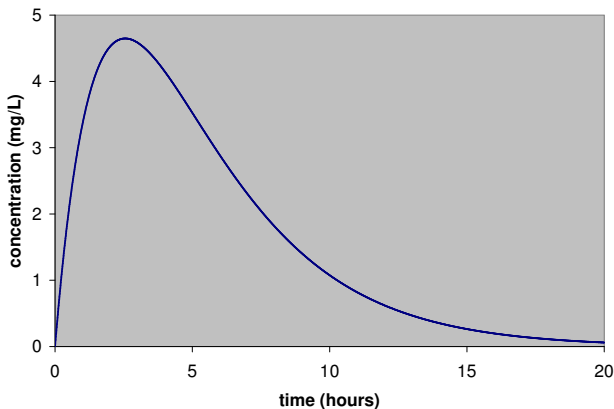
Differential equation models: e.g. 'one-compartment' pharmacokinetic model



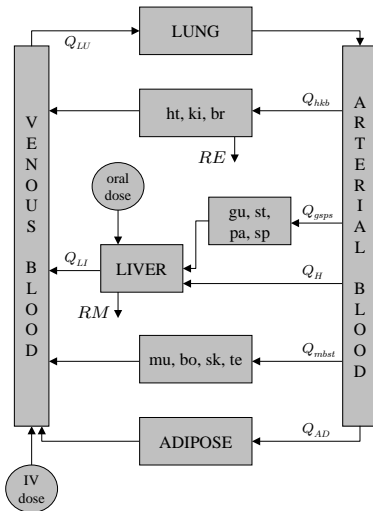
$$\begin{aligned}\frac{dA_g}{dt} &= -k_a A_g & \frac{dA_b}{dt} &= k_a A_g - k_e A_b \\ A_g(0) &= Dose & A_b(0) &= 0\end{aligned}$$

One-compartment model, solution

$$C_b(t) = \frac{Dose}{V_b} \times \frac{k_a}{k_e - k_a} \{ \exp(-k_a t) - \exp(-k_e t) \}$$



Physiologically based PK model



$$\frac{dA_{LU}}{dt} = Q_{LU} \times (C_{VEN} - C_{LU}/K_{PLU})$$

$$\frac{dA_{hkb}}{dt} = Q_{hkb} \times (C_{ART} - C_{hkb}/K_{Phkb}) - RE(t)$$

$$\frac{dA_{gsps}}{dt} = Q_{gsps} \times (C_{ART} - C_{gsps}/K_{Pgsps})$$

$$\frac{dA_{LI}}{dt} = Q_H \cdot C_{ART} + Q_{gsps} \cdot C_{gsps}/K_{Pgsps} + RA(t) - Q_{LI} \cdot C_{LI}/K_{PLI} - RM(t)$$

$$\frac{dA_{mbst}}{dt} = Q_{mbst} \times (C_{ART} - C_{mbst}/K_{Pmbst})$$

$$\frac{dA_{AD}}{dt} = Q_{AD} \times (C_{ART} - C_{AD}/K_{PAD})$$

$$\frac{dA_{ART}}{dt} = Q_{LU} \cdot C_{LU}/K_{PLU} - CO \cdot C_{ART}$$

$$\frac{dA_{VEN}}{dt} = Q_{hkb} \cdot C_{hkb}/K_{Phkb} + Q_{LI} \cdot C_{LI}/K_{PLI} + Q_{mbst} \cdot C_{mbst}/K_{Pmbst} + Q_{AD} \cdot C_{AD}/K_{PAD} + RI(t) - Q_{LU} \cdot C_{VEN}$$

BUGS language specification

$$\frac{dA_g}{dt} = -k_a A_g \quad \frac{dA_b}{dt} = k_a A_g - k_e A_b \quad [A_g(0) = Dose, \quad A_b(0) = 0]$$

```

model {
  for (i in 1:N) {
    y[i] ~ dnorm(Cb[i], tau)
    Cb[i] <- solution[i, 2] / Vb
  }
  solution[1:N, 1:2] <- ode(init[], grid[], D(A[1:2], t), origin, tol)
  D(A[1], t) <- -ka * A[1]
  D(A[2], t) <- ka * A[1] - ke * A[2]
#  solution[1:N, 1:2] <- one.comp(init[], grid[], theta[], origin, tol)
#  theta[1] <- ka; theta[2] <- ke
  init[1] <- dose; init[2] <- 0
  ke ~ dunif(0, 10)
  ka ~ dunif(0, 10)
  Vb ~ dunif(0, 1000)
  sd ~ dunif(0, 10)
  tau <- 1 / pow(sd, 2)
}

```

BUGS language specification

$$\frac{dA_g}{dt} = -k_a A_g \quad \frac{dA_b}{dt} = k_a A_g - k_e A_b \quad [A_g(0) = Dose, \quad A_b(0) = 0]$$

```

model {
  for (i in 1:N) {
    y[i] ~ dnorm(Cb[i], tau)
    Cb[i] <- solution[i, 2] / Vb
  }
#  solution[1:N, 1:2] <- ode(init[], grid[], D(A[1:2], t), origin, tol)
#  D(A[1], t) <- -ka * A[1]
#  D(A[2], t) <- ka * A[1] - ke * A[2]
  solution[1:N, 1:2] <- one.comp(init[], grid[], theta[], origin, tol)
  theta[1] <- ka; theta[2] <- ke
  init[1] <- dose; init[2] <- 0
  ke ~ dunif(0, 10)
  ka ~ dunif(0, 10)
  Vb ~ dunif(0, 1000)
  sd ~ dunif(0, 10)
  tau <- 1 / pow(sd, 2)
}

```

Source code (template)

```

MODULE WBDiffOneComp;

IMPORT
  WBDiffODEMath, Math;

TYPE
  Equations = POINTER TO RECORD (WBDiffODEMath.Equations) END;
  Factory = POINTER TO RECORD (WBDiffODEMath.Factory) END;

CONST
  nEq = 2;

VAR
  fact:- WBDiffODEMath.Factory;

PROCEDURE (e: Equations) Derivatives (IN theta, A: ARRAY OF REAL;
                                       n: INTEGER; t: REAL;
                                       OUT dAdt: ARRAY OF REAL);

VAR
  ka, ke: REAL;
BEGIN
  ka := theta[0]; ke := theta[1];
  dAdt[0] := -ka * A[0];
  dAdt[1] := ka * A[0] - ke * A[1];
END Derivatives;

.....

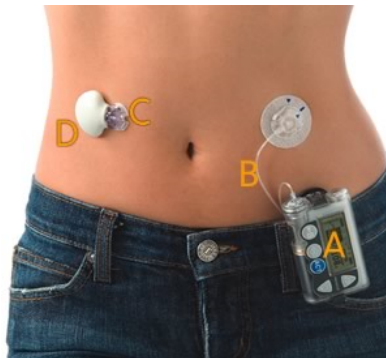
```

Source code continued

```
PROCEDURE (e: Equations) Derivatives (IN theta, A: ARRAY OF REAL;  
                                         n: INTEGER; t: REAL;  
                                         OUT dAdt: ARRAY OF REAL);  
  
VAR  
    ka, ke: REAL;  
BEGIN  
    ka := theta[0]; ke := theta[1];  
    dAdt[0] := -ka * A[0];  
    dAdt[1] := ka * A[0] - ke * A[1];  
END Derivatives;
```


Forced dynamical systems, e.g. continuous glucose monitoring (CGM)

- ▶ Traditional therapy for Type 1 diabetes: multiple insulin injections informed by self-monitoring of blood glucose (BG) with finger-stick, say.
- ▶ Artificial pancreas: subcutaneous sensor measures glucose continuously and relays information to insulin pump controller



CGM continued: Kinetic model

- ▶ Price paid for *continuous* monitoring is that we measure BG indirectly – we actually monitor *interstitial* glucose (IG)
- ▶ Need to characterise relationship between BG and IG...
- ▶ Simple compartmental model:

$$\frac{dIG(t)}{dt} = -\theta\{IG(t) - BG(t)\}$$

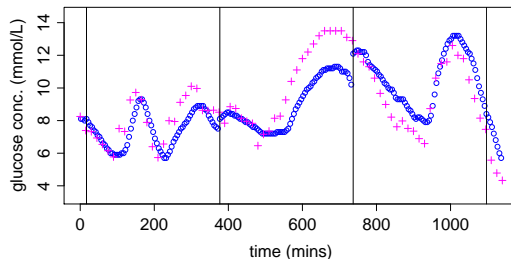
- ▶ Have no model for $BG(t)$ as it is driven by external factors. Instead we observe BG at a grid of time points
- ▶ But to solve the equation we need to be able to evaluate BG at *any* time
- ▶ \Rightarrow interpolate between the observations??

Forcing functions

- ▶ Interpolated BG series is known as a *forcing function*
- ▶ Occur in many settings, e.g.
 - ▶ ambient temperature or pressure
 - ▶ wind stress/speed
 - ▶ light intensity or availability of food in ecological modelling
 - ▶ insulin or glucose levels in diabetes research

Data

- ▶ Sensor actually measures interstitial *current* \Rightarrow needs calibrating (use recently observed BG values; every six hours)



- ▶ We assume calibration comprises scaling (factor F) and shifting ($+B$)
- ▶ 12 individuals over 19 hours; sensor glucose (SG) measured every 5 mins ($n = 228$) and BG measured every 15 mins ($m = 77$); 5 'calibration periods'

Statistical model

$$SG_{ij} \sim N(CIG_{ij}, \sigma_{iP[i,j]}^2), \quad CIG_{ij} = F_{iP[i,j]} \times IG_{ij} + B_{iP[i,j]}$$

- ▶ SG = sensor glucose; CIG = calibrated IG
- ▶ F = calibration scale factor; B = calibration shift
- ▶ $P[i,j]$ = calibration period to which SG_{ij} belongs
- ▶ IG_{ij} = solution to ODE at time t_{ij} , subject to current values of θ and initial condition $IG_0 = IG(t = t_0)$
- ▶ Normal population distributions for $\log \theta_i$, $\log IG_{0i}$, $\log F_{ik}$, B_{ik} and $\log \sigma_{ik}$, $i = 1, \dots, 12$, $k = 1, \dots, 5$
- ▶ Normal priors for population means and log-SDs

BUGS code (likelihood + exchangeability)

```

for (i in 1:N) {
  for (j in 1:n) {
    SG[i, j] ~ dnorm(mean[i, j], prec[i, j])
    mean[i, j] <- F[i, P[i, j]] * IG[i, j] + B[i, P[i, j]]
    prec[i, j] <- 1/pow(sigma[i, P[i, j]], 2)
  }
  IG[i, 1:n] <- glucose(eta[i], x[i, 1:n], theta[i], psi[i], origin[i], tol)
  psi[i] <- diff.interp(s[1:nu], BG[i, 1:nu])

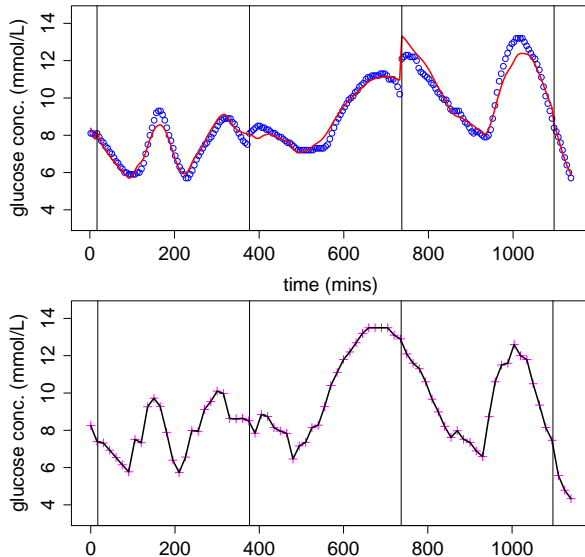
  log(eta[i]) <- log.eta[i]
  log.eta[i] ~ dnorm(mu.eta, tau.eta)
  log(theta[i]) <- log.theta[i]
  log.theta[i] ~ dnorm(mu.theta, tau.theta)
  for (j in 1:5) {
    log(sigma[i, j]) <- log.sigma[i, j]
    log.sigma[i, j] ~ dnorm(mu.sigma, tau.sigma)
    log(F[i, j]) <- C[i, j, 1]
    B[i, j] <- C[i, j, 2]
    C[i, j, 1:2] ~ dmnorm(mu.C[], Sigma.inv[,])
  }
}

```

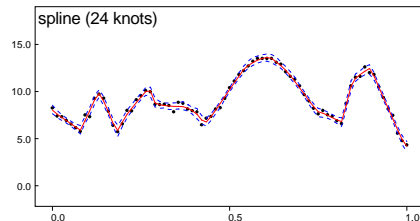
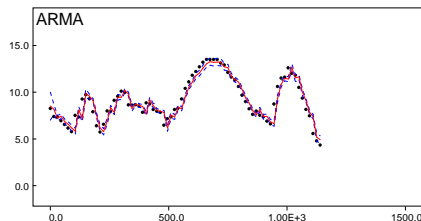
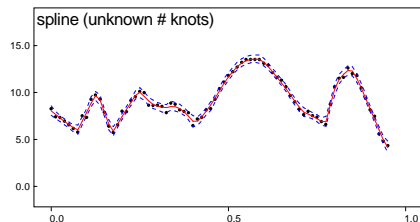
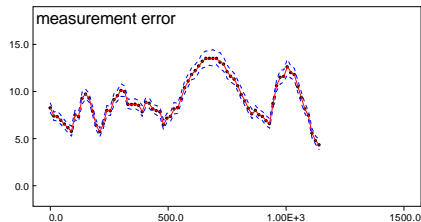
BUGS code continued (priors)

```
tau.sigma <- 1/pow(omega.sigma, 2)
tau.eta <- 1/pow(omega.eta, 2)
tau.theta <- 1/pow(omega.theta, 2)
Sigma[1:2, 1:2] <- inverse(Sigma.inv[,])
log(omega.sigma) <- log.omega.sigma
log(omega.eta) <- log.omega.eta
log(omega.theta) <- log.omega.theta
mu.sigma ~ dnorm(0, 0.0001)
mu.eta ~ dnorm(0, 0.0001)
mu.theta ~ dnorm(0, 0.0001)
mu.C[1:2] ~ dmnorm(zero[,], prior.prec.C[,])
log.omega.sigma ~ dnorm(0, 0.0001)
log.omega.eta ~ dnorm(0, 0.0001)
log.omega.theta ~ dnorm(0, 0.0001)
Sigma.inv[1:2, 1:2] ~ dwish(R[,], 2)
```

Results (1)



Models for forcing data...

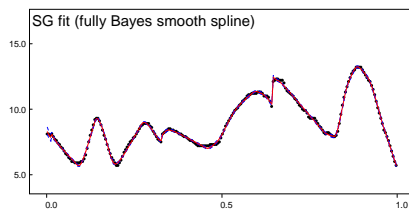
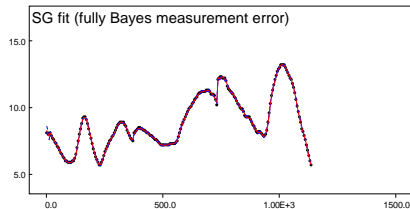
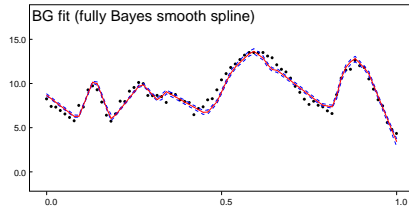
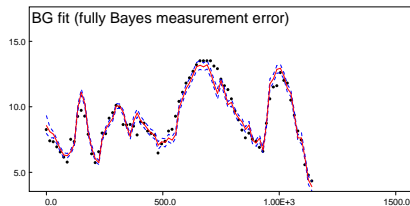


Implementation

- ▶ Forcing functions are non-standard nodes: deterministic but not 'logical nodes' – logical nodes are functions of other nodes in the graph – forcing functions depend on dummy variable of integration, which does not belong in the graph...
- ▶ We may fit a model to a set of observations at specific times but when that model is fed into the ODE system it needs to be defined $\forall t$
- ▶ We have defined a new class of functions that are only fully defined when coupled to an ODE solver... Allows specification of *smooth* as well as *uncertain* forcing functions

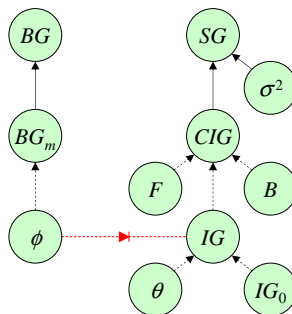
```
IG[1:n] ← glucose(init, grid[], par[], origin, tol, ff)
ff ← diff.pwpoly(phi[], 0, 1, 1)
ff ← diff.interp(grid.BG[], fitted.BG[])
ff ← diff.step(change.points[], levels[])
```

Results (2): Fully Bayesian analysis: Plausible??



Cutting feedback

- ▶ We can acknowledge uncertainty in the forcing function without allowing the fit to be compromised by inappropriate ‘tweaking’ of the response model
- ▶ Introduce valve between modelled forcing function and ODE...
 ϕ acts as a parent of SG but SG not a child of ϕ



Cutting feedback continued

- ▶ 'Bayesian' analogue of two-stage analysis
- ▶ Implement via 'cut()' function and/or 'dpick()' distribution

$ff \leftarrow \text{diff.pwpoly}(\text{phi}[], \dots)$

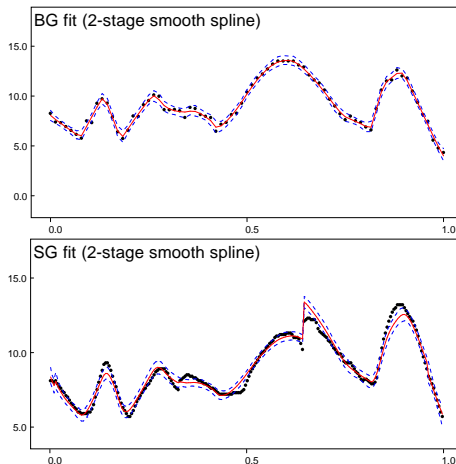
one-stage: $\text{phi}[j] \leftarrow \text{cut}(\text{phi.bg}[j])$

two-stage: $\text{phi.bg}[] \sim \text{dpick}(\langle \text{posterior sample from BG analysis} \rangle)$

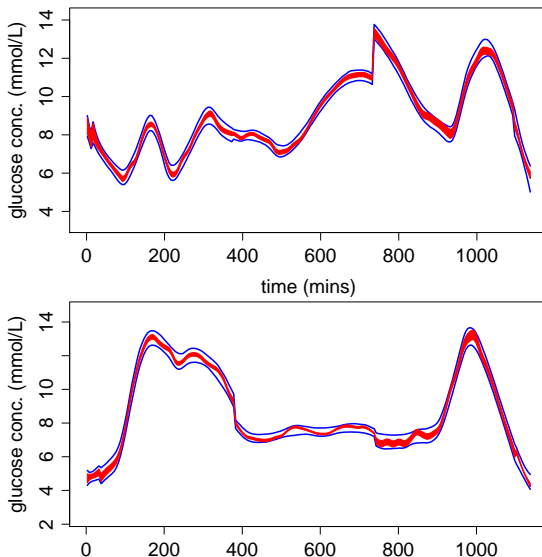
- ▶ phi.bg represents parameters from analysing BG data alone
- ▶ Allows BG model to be faithful to features in forcing data without over-emphasising 'belief' in response model

Results (3): No feedback...

- Acknowledges uncertainty without response model being tweaked...



Uncertainty propagation...



Conclusions + future work

- ▶ Dynamical systems can be modelled using standard graphical modelling theory – provides great scope for dealing with complexities of ‘real data’, e.g. error distributions, arbitrary model structures, ...
 - ▶ But: specification limited in BUGS; not available in JAGS (yet); MCSim??; other packages emerging...
 - ▶ Need to be careful with priors as ODE solvers can be sensitive to inputs – work in progress
 - ▶ Future: implementation in OpenBUGS (www.openbugs.info)
- ▶ New graphical node types allow uncertainty propagation and smoothness constraints
 - ▶ Same ideas allow (smoothly) time-varying parameters – could infer shape of circadian rhythms, say
- ▶ Fully Bayesian analysis may be inappropriate
 - ▶ Diagnostics for conflict
 - ▶ Methods for two-stage sampling

Acknowledgements

- ▶ Glucose data: Many thanks to Roman Hovorka *et al.*, Diabetes Modelling Group, University Department of Paediatrics, University of Cambridge, UK
- ▶ Thanks to AstraZeneca for initially funding WBDiff development
- ▶ Thanks to Andrew Thomas, Chen Wei (BSU) and Steve Miller (BSU) for their work on the project
- ▶ Thanks to David Spiegelhalter (BSU), Martyn Plummer (IARC), Dani De Angelis (BSU) and Anne Presanis (BSU) for very helpful discussions