1.1. Applying the Laplace approximation to the Generalized Kalman Filter – with an application to Stochastic Volatility Models

Let y_i be an N dimensional multivariate time series for $i = 1, \ldots, n$ where y_i is a random vector with probability density function $p(y_i|\alpha_i)$. For each i, the α_i are random vectors which satisfy the condition

$$\alpha_i = T_i(\alpha_{i-1}, y_{i-1}) + \eta_i$$

. where $\mu_{\eta_i} = 0$ and $\sigma^2_{\eta_i} = \sigma^2_\eta$.

Let $p(\alpha_1)$ be the probability density function for α_1 before y_1 is oberved. After observing y_1 we want to calculate the probability distribution of α_1 given y_1. This is given by

$$p(\alpha_1|y_1) = p(y_1|\alpha_1)p(\alpha_1)/p(y_1) \tag{1.1}$$

where

$$p(y_1) = \int_{-\infty}^{\infty} p(y_1|\alpha_1)p(\alpha_1)\,d\alpha_1 \tag{1.2}$$

let $\phi(y_1, \alpha_1) = \log(p(y_1|\alpha_1)p(\alpha_1))$ Let $\hat\alpha_1(y_1) = \max_{\alpha_1}\{\phi(y_1, \alpha_1)\}$. Approximate ϕ by its second order taylor expansion in α_1 at $\hat\alpha_1$.

$$\phi(y_1, \alpha_1) \approx \phi(y_1, \hat\alpha_1) + D^2_{\alpha_1\alpha_1}\phi(y_1, \hat\alpha_1(y_1))(\alpha_1 - \hat\alpha_1(y), \alpha_1 - \hat\alpha_1(y))$$

so that

$$p(y) \approx e^{\phi(y_1,\hat\alpha_1(y_1))}\int_{-\infty}^{\infty}\exp\Big\{ -\big(-D^2_{\alpha_1\alpha_1}\phi(y_1, \hat\alpha_1(y_1))(\alpha_1 - \hat\alpha_1(y), \alpha_1 - \hat\alpha_1(y))\big)\Big\}\,d\alpha_1$$

Making a change of variables and integrating we obtain

$$p(y_1) \approx e^{\phi(y_1,\hat\alpha_1(y_1))}(2\pi)^{n/2}|-D^2_{\alpha_1\alpha_1}\phi(y_1, \hat\alpha_1(y_1)))|^{-1/2} \tag{1.4}$$

This is the Laplace approximation to the integral in (1.2).

To calculate (1.4) it is necessary to maximize $\phi(y_1, \alpha_1)$ with respect to α_1 and to calculate its hessian matrix with respect to α_1.

For the maximization we employ the Newton-Raphson algorithm. Let $\beta_0 = \mu_{\alpha_1}$

$$\beta_{i+1} = \beta_i - \big\{D^2_{\alpha_1\alpha_1}\phi(y_1, \beta_i)\big\}^{-1}(D_{\alpha_1}\phi(y_1, \beta_i))$$

This operation is carried out a fixed number r times and then $\hat\alpha_1(y_1) \approx \beta_r$. For "well behaved" problems the sequence β_i converges quadratically to $\hat\alpha_1(y_1)$. We approximate $p(\alpha_1|y_1)$ by a multivariate normal with

$$\begin{aligned}\mu_{\alpha_1|y_1} &= \beta_r\\ \sigma^2_{\alpha_1|y_1} &= \big\{-D^2_{\alpha_1\alpha_1}\phi(y_1, \beta_r)\big\}^{-1}\end{aligned}$$

and approximate $p(\alpha_2|y_1)$ by a multivariate normal with

$$\begin{aligned}\mu_{\alpha_2|y_1} &= T(\beta_r, y_1)\\ \sigma^2_{\alpha_2|y_1} &= D_{\alpha_1}T(\beta_r, y_1)\sigma^2_{\alpha_1|y_1}D_{\alpha_1}T(\beta_r, y_1)' + \sigma^2_\eta\end{aligned}$$

Now

$$p(y_2|y_1) = \int_{-\infty}^{\infty} p(y_2|\alpha_2)p(\alpha_2|y_1)\,d\alpha_2 \tag{1.5}$$

As above we maximize the integrand of (1.5) with respect to α_2 and use the Laplace approximation to the integral. This produces the sequence of conditional probabilities, $p(y_i|y_{i-1})$. The log-likelihood function for the observed sequence y_i is given by

$$\sum_{i=1}^{n}\log\big(p(y_i|y_{i-1})\big) \tag{1.6}$$

1.2. Parameter estimation

Although we have not explicitly shown them the conditional likelihood functions $p(y_i|y_{i-1})$ depend on a number of parameters. These parameters include the specification of T, other parameters in the probability density $p(y_i|\alpha_i)$ and parameters which determine σ^2_η. If we denote these parameters by θ and write $\big(p(y_i|y_{i-1},\theta)\big)$ to indicate this dependence the log-likelihood function becomes

$$\sum_{i=1}^{n} \log\big(p(y_i|y_{i-1},\theta)\big) \tag{1.7}$$

the maximum likelihood estimates for the parameter vector θ are found by maximizing (1.7) with respect to θ.

1.3. The stochastic volatility model

The version of the stochastic volatility model presented here is from the paper Multivariate Stochastic Volatility Models: Estimation and a comparison with VGARCH Models by Danielsson.

It is assumed that y_i has a multivariate normal distribution with $\mu_{y_i}=0$ and covariance matrix $\Omega_i(\alpha_i) = H_i(\alpha_i)RH_i(\alpha_i)$ where $H_i(\alpha_i)$ is an $m\times m$ diagonal matrix whose j'th element on the diagonal is given by $\exp(\alpha_{ij})/2$ where the α_{ij} satisfy the relationship

$$\alpha_i = w + \text{elem\_prod}(\delta,\alpha_{i-1}) + \text{elem\_prod}(\lambda_1,y_{i-1}) + \text{elem\_prod}(\lambda_2,|y_{i-1}|) + \eta_i$$

where η_i is a multivariate normal random variable with $\mu_{\eta_i}=0$ and $\sigma^2_{\eta_i}=\sigma^2_\eta$. If u and v are two vectors with j'th component u_j and v_j elem_prod(i,v) is the vector with j'th componment u_jv_j. R is an $m\times m$ postive definite matrix satisfying $r_{jj}=1$, that is a corellation matrix. Then

$$\log\big(p(y_i|\alpha_i)\big) = -0.5\log|\Omega_i(\alpha_i)| - 0.5y_i'\Omega_i(\alpha_i)^{-1}y_i$$

and the distribution of $\alpha_i|y_{i-1}$ is multivariate normal with mean vector and covariance matrix given by

$$\begin{aligned}\mu_{\alpha_i|y_{i-1}} &= w + \text{elem\_prod}(\delta,\mu_{\alpha_{i-1}|y_{i-1}}) + \text{elem\_prod}(\lambda,y_{i-1})\\ \sigma^2_{\alpha_i|y_{i-1}} &= \text{diag}(\delta)\sigma^2_{\alpha_{i-1}|y_{i-1}}\text{diag}(\delta) + \sigma^2_\eta\end{aligned}$$

diag(δ) is the diagonal matrix whose diagonal is equal to the vector δ.

$$\begin{aligned}\log\big(p(y_i|\alpha_i)p(\alpha_i|y_{i-1})\big) = &-0.5\log|\Omega_i(\alpha_i)| - 0.5y_i'\Omega_i(\alpha_i)^{-1}y_i - 0.5\log|\sigma^2_{\alpha_i|y_{i-1}}|\\ &- 0.5(\alpha_i-\mu_{\alpha_i|y_{i-1}})'(\sigma^2_{\alpha_i|y_{i-1}})^{-1}(\alpha_i-\mu_{\alpha_i|y_{i-1}})\end{aligned} \tag{1.8}$$

To perform the Newton-Raphson calculations it is necessary to calculate the first and second derivatives of expression (1.8) with respect to the parameter vector α. This is the most involved part of the calculations and will depend on the particular form of the model. In the present case the calculations are simplified by the fact that Ω_i only depends on α through the diagonal matrix $H(\alpha_i)$.

The probability density function $p(\alpha_1)$ is assumed to be multivariate normal with $\mu_{\alpha_1}=\theta_0$ and $\sigma^2_{\alpha_1}=0$.

1.4. The Data

The data consist of the daily Mark/Dollar and Yen/dollar exchange rates and the US and Japaneese stock index data. There are 1301 time periods with some missing data. The missing data which are denoted by the impossibly large value of 10,000 were replaced with the average from the period before and after. They can howerever easily be estimated in the model is desired.

1.5. The Results

The model was fit with various combinations of the parameters and the log-likelihood was examined to investigate the improvement in fit due to the addition of the parameters.

Parameters in model	number of parameters	log-likelihood
$w, \delta, R, \sigma_\eta^2$	24	3774.7
$w, \delta, R, \sigma_\eta^2, \lambda_1$	28	3806.6
$w, \delta, R, \sigma_\eta^2, \lambda_1, \theta_0$	32	3808.6
$w, \delta, R, \sigma_\eta^2, \lambda_1, \theta_0, \lambda_2$	36	3811.2

The parameters θ_0 and λ_2 did not produce a significant improvement to the fit. λ_2 measures the asymmetry in the response of the variance to positive and negative shocks.

Here are the parameter estimates and their standard deviations for the model with $w, \delta, R, \sigma_\eta^2$, and λ_1.

```
index   name    value      std dev
   1   w(1)       -1.3749e-001 4.9434e-002
   2   w(2)       -6.5649e-001 1.6161e-001
   3   w(3)        3.1693e-002 1.0574e-002
   4   w(4)       -1.2973e-002 1.5375e-002
   5   lambda1(1)  1.5564e-001 4.9688e-002
   6   lambda1(2)  1.8647e-001 6.9525e-002
   7   lambda1(3)  -6.9265e-002 1.4158e-002
   8   lambda1(4)  -1.6689e-001 3.1626e-002
   9   delta(1)    8.2229e-001 4.6074e-002
  10   delta(1)    5.0848e-001 1.0785e-001
  11   delta(1)    9.5763e-001 1.4602e-002
  12   delta(1)    9.3610e-001 1.8812e-002
  29   R(1,1)      1.0000e+000 0.0000e+000
  30   R(1,2)      5.3821e-001 2.2883e-002
  31   R(1,3)      -7.1704e-002 2.9477e-002
  32   R(1,4)      -3.8796e-002 2.9278e-002
  33   R(2,1)      5.3821e-001 2.2883e-002
  34   R(2,2)      1.0000e+000  0.0000e+000
  35   R(2,3)      -1.2932e-001 2.9111e-002
  36   R(2,3)      -4.1466e-002 2.9468e-002
  37   R(3,1)      -7.1704e-002 2.9477e-002
  38   R(3,2)      -1.2932e-001 2.9111e-002
  39   R(3,3)      1.0000e+000  0.0000e+000
  40   R(1,4)      8.8811e-002 2.9085e-002
  41   R(4,1)      -3.8796e-002 2.9278e-002
  42   R(4,2)      -4.1466e-002 2.9468e-002
  43   R(4,3)      8.8811e-002 2.9085e-002
  44   R(4,4)      1.0000e+000  0.0000e+000
  45   Omega(1,1)  6.5973e-001 6.3099e-002
  46   Omega(1,2)  1.9827e-001 1.6129e-002
  47   Omega(1,3)  -1.3395e-001 5.4982e-002
  48   Omega(1,4)  -3.5161e-002 2.6676e-002
  49   Omega(2,1)  1.9827e-001 1.6129e-002
  50   Omega(2,2)  2.0570e-001 2.3994e-002
  51   Omega(2,3)  -1.3489e-001 3.2608e-002
  52   Omega(2,4)  -2.0985e-002 1.5016e-002
  53   Omega(3,1)  -1.3395e-001 5.4982e-002
  54   Omega(3,2)  -1.3489e-001 3.2608e-002
  55   Omega(3,3)  5.2895e+000 5.7872e-001
  56   Omega(3,4)  2.2791e-001 7.9318e-002
  57   Omega(4,1)  -3.5161e-002 2.6676e-002
  58   Omega(4,2)  -2.0985e-002 1.5016e-002
  59   Omega(4,3)  2.2791e-001 7.9318e-002
  60   Omega(4,4)  1.2451e+000 1.7043e-001
  61   Z(1,1)      2.3967e-001 7.4268e-002
  62   Z(1,2)      2.0711e-001 5.5599e-002
  63   Z(1,3)      3.8832e-002 1.8505e-002
  64   Z(1,4)      2.4097e-002 2.0344e-002
  65   Z(2,1)      2.0711e-001 5.5599e-002
  66   Z(2,2)      4.6309e-001 1.1143e-001
  67   Z(2,3)      3.4298e-002 2.3017e-002
  68   Z(2,4)      9.6831e-003 2.9999e-002
```

```
  69   Z(3,1)      3.8832e-002 1.8505e-002
  70   Z(3,2)      3.4298e-002 2.3017e-002
  71   Z(3,3)      3.9101e-002 1.6885e-002
  72   Z(3,4)      2.4602e-002 1.1053e-002
  73   Z(4,1)      2.4097e-002 2.0344e-002
  74   Z(4,2)      9.6831e-003 2.9999e-002
  75   Z(4,3)      2.4602e-002 1.1053e-002
  76   Z(4,4)      9.6109e-002 3.4268e-002
```

The AD Model Builder TPL file for the model is given below.

```
DATA_SECTION
  init_int ndim
  init_int nobs
  int ndim1
  int ndim2
 !! ndim1=ndim*(ndim+1)/2;
 !! ndim2=ndim*(ndim-1)/2;
  init_matrix Y(1,nobs,1,ndim)
 LOC_CALCS
  // replace missing values (10000) with the average of before and after.
  for (int i=2;i<nobs;i++)
    for (int j=1;j<=ndim;j++)
      if (Y(i,j)==10000)
      {
        int i2=i+1;
        do
        {
          if (Y(i2,j)==10000)
            i2++;
          else
            break;
        }
        while(1);
        Y(i,j)=(Y(i-1,j)+Y(i2,j))/2.;
        if (Y(i,j)>100.0)     // did this work
          cerr << " Y(i,j) too big " << Y(i,j) << endl;
      }
 END_CALCS

PARAMETER_SECTION
  matrix h_mean(1,nobs,1,ndim)
  3darray h_var(1,nobs,1,ndim,1,ndim)
  number ldR;
  init_vector theta0(1,ndim,3);
  vector lmin(1,nobs)
  init_bounded_vector w(1,ndim,-10,10)
  vector w1(1,ndim)
  init_vector lambda(1,ndim,2)
  init_vector lambda2(1,ndim,-1)
  init_bounded_vector delta(1,ndim,0,.98)
  sdreport_matrix R(1,ndim,1,ndim)
  sdreport_matrix Omega(1,ndim,1,ndim)
  matrix ch_R(1,ndim,1,ndim)
  matrix Rinv(1,ndim,1,ndim)
  init_bounded_vector v_R(1,ndim2,-1.0,1.0)
  sdreport_matrix Z(1,ndim,1,ndim)
  matrix ch_Z(1,ndim,1,ndim)
  init_bounded_vector v_Z(1,ndim1,-1.0,1.0)
  matrix S(1,ndim,1,ndim);
  objective_function_value f
INITIALIZATION_SECTION
  delta 0.9
PROCEDURE_SECTION

  fill_the_matrices();
  int sgn;
  ldR=ln_det(R,sgn);
  Rinv=inv(R);
  dvar_vector tmp(1,ndim);
  dvar_matrix sh(1,ndim,1,ndim);
  h_mean(1)=theta0;
  h_var(1)=0;
  for (int i=2;i<=nobs;i++)
  {
    dvar_vector tmean=update_the_means(w,h_mean(i-1),Y(i-1));
    dvar_matrix v=update_the_variances(h_var(i-1));
    tmp=tmean;
    dvar_vector h(1,ndim);
    dvar_vector gr(1,ndim);
```

```
    for (int ii=1;ii<=4;ii++)  // do the Newton-Raphson 4 times
    {
      xfp12(tmp, Y(i),tmean,v,gr,sh); // get 1st and 2nd derivatives
      h=-solve(sh,gr);  //sh is hessian and gr is the gradient
      tmp+=h;  // add new step h
    }
    double nh=norm2(value(h)); // check size of h for convergence
    if (nh>1.e-1)
      cout << "No convergence in NR " << nh << endl;
    if (nh>1.e+02)
    {
      f+=1.e+7;   // this ensures that the function minimizer will take a
      return;    // smaller step
    }
    h_mean(i)=tmp;
    h_var(i)=inv(sh);
    lmin(i)=fp(tmp,Y(i),tmean,v);
    int sgn;
    f+=lmin(i)+0.5*ln_det(sh,sgn);  // Laplace approximation
  }
  f-=0.5*nobs*ndim*log(2.*3.14159);
  Omega=S;

FUNCTION  dvar_vector update_the_means(dvar_vector& w,dvar_vector& m,dvector& e)
  dvar_vector tmp= w+elem_prod(delta,m)+elem_prod(lambda,e);
  if (active(lambda2))
    tmp+=elem_prod(lambda2,fabs(e));
  return tmp;

FUNCTION  dvar_matrix update_the_variances(dvar_matrix& v)
  dvar_matrix tmp(1,ndim,1,ndim);
  for (int i=1;i<=ndim;i++)
  {
    for (int j=1;j<=i;j++)
    {
      tmp(i,j)=delta(i)*delta(j)*v(i,j);
      if (i!=j) tmp(j,i)=tmp(i,j);
    }
  }
  tmp+=Z;
  return tmp;

FUNCTION dvariable fp(dvar_vector& h, dvector& y, dvar_vector& m,dvar_matrix& v)
  dvar_vector eh=exp(.5*h);
  for (int i=1;i<=ndim;i++)
  {
    for (int j=1;j<=i;j++)
    {
      S(i,j)= eh(i)*eh(j)*R(i,j);
      if (i!=j) S(j,i)=S(i,j);
    }
  }

  dvariable lndet;
  dvariable sgn;
  dvar_vector u=solve(S,y,lndet,sgn);
  dvariable l;
  l=.5*lndet+.5*(y*u);
  dvar_vector hm=h-m;
  w1=solve(v,hm,lndet,sgn);
  l+=.5*lndet+.5*(w1*hm);
  return l;

FUNCTION void xfp12(dvar_vector& h, dvector& y,dvar_vector& m,dvar_matrix& v, dvar_vector
gr,dvar_matrix& hess)
  dvar_vector ehinv=exp(-.5*h);
  dvariable lndet;
  dvariable sgn;
  dvar_vector ys=elem_prod(ehinv,y);
  dvar_vector u=Rinv*ys;
  gr=0.5;
  dvar_vector vv=elem_prod(ys,u);
  gr-=.5*vv;
  dvar_vector hm=h-m;
  dvar_vector w=solve(v,hm,lndet,sgn);
  gr+=w;
  for (int i=1;i<=ndim;i++)
  {
    for (int j=1;j<=i;j++)
    {
      hess(i,j)=0.25*ys(i)*ys(j)*Rinv(i,j);
```

```
      if (i!=j) hess(j,i)=hess(i,j);
    }
  }
  for (i=1;i<=ndim;i++)
  {
    hess(i,i)+=.25*vv(i);
  }
  hess+=inv(v);

FUNCTION  fill_the_matrices
  int ii=1;
  ch_Z.initialize();
  for (int i=1;i<=ndim;i++)
  {
    for (int j=1;j<=i;j++)
      ch_Z(i,j)=v_Z(ii++);
    ch_Z(i,i)+=0.5;
  }
  Z=ch_Z*trans(ch_Z);
  ch_R.initialize();
  ii=1;
  for (i=1;i<=ndim;i++)
  {
    for (int j=1;j<i;j++)
      ch_R(i,j)=v_R(ii++);
    ch_R(i,i)+=0.1;
    ch_R(i)/=norm(ch_R(i));
  }
  R=ch_R*trans(ch_R);

REPORT_SECTION
  report<<"observed"<<Y<<endl;
  for (int i=1;i<=nobs;i++)
  {
    report<< "mean" <<endl;
    report<< h_mean(i) <<endl;
    report<< "covariance" <<endl;
    report<<h_var(i)<<endl;
    report<<endl;
  }
  report<< "S(nobs) " << endl;
  report<< Omega << endl;
  report<< "Z " << endl;
  report<< Z << endl;
  report<< "R " << endl;
  report<< R << endl;

TOP_OF_MAIN_SECTION
  arrmblsize=20000000;
  gradient_structure::set_CMPDIF_BUFFER_SIZE(25000000);
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(1000000);
```