



ภาคผนวก

ภาษา **Visual Basic 6.0** ที่ใช้ในการสร้างตัวแปรสุ่มที่มีการแจกแจงแบบแกมมา (**Gamma Distribution**)  
และตัวแปรสุ่มที่มีการแจกแจงแบบทวินามลบ (**Negative Binomial Distribution**) ด้วยวิธีการแปลง  
ผกผัน วิธีการยอมรับและปฏิเสธและวิธีการรวม

Option Explicit

Public a, b, z, answer, answer\_1, x1, s, sample, loop\_test, sss As Variant

Public k, k\_1, a5, a5\_1, w, w\_1, i, j, select\_define\_value As Variant

Public expo, x\_expo, sum\_expo, random\_gamma, sum\_gamma, m\_gamma As Variant

Public m\_gamma\_acc\_rej As Variant

Public dif\_x\_mean, sum\_for\_var, var\_gamma, standard\_dev, estimator\_alpha, estimator\_beta As Variant

Public dif\_est\_alpha, sum\_dif\_est\_alpha, dif\_est\_beta, sum\_dif\_est\_beta As Variant

Public dif\_xx\_mean, sum\_for\_var\_acc\_rej, var\_gamma\_acc\_rej, standard\_dev\_acc\_rej As Variant

Public estimator\_alpha\_acc\_rej, estimator\_beta\_acc\_rej As Variant

Public dif\_est\_alpha\_acc\_rej, sum\_dif\_est\_alpha\_acc\_rej As Variant

Public dif\_est\_beta\_acc\_rej, sum\_dif\_est\_beta\_acc\_rej As Variant

Public mse\_alpha\_acc\_rej, rmse\_alpha\_acc\_rej, mse\_beta\_acc\_rej, rmse\_beta\_acc\_rej As Variant

Public alpha, beta, mse\_alpha, rmse\_alpha, mse\_beta, rmse\_beta, neg\_r\_v As Variant

Public number\_trial, x\_success, prob\_success, geo\_random, neg\_random As Variant

Public p\_estimator, dif\_p, dif\_pp, sum\_dif\_p As Variant

Public mse\_neg, rmse\_neg As Variant

Public aa, bb, qq, zeta, dd, vv, yy, zz, ww, sum\_xx As Variant

Public sum\_neg\_r\_v, mean\_neg\_for\_estimate, mean\_estimate As Variant

Public est\_mean\_gamma, est\_variance\_gamma, sum\_est\_mean\_gamma, sum\_est\_variance\_gamma As Variant

Public mse\_mean\_gamma, rmse\_mean\_gamma, mse\_variance\_gamma, rmse\_variance\_gamma As Variant

Public est\_mean\_gamma\_acc\_rej, est\_variance\_gamma\_acc\_rej, sum\_est\_mean\_gamma\_acc\_rej, sum\_est\_variance\_gamma\_acc\_rej As Variant

Public mse\_mean\_gamma\_acc\_rej, rmse\_mean\_gamma\_acc\_rej, mse\_variance\_gamma\_acc\_rej, rmse\_variance\_gamma\_acc\_rej As Variant

Public dif\_neg, sum\_for\_var\_neg, var\_neg, mse\_var\_neg, rmse\_var\_neg As Variant

Public dif\_for\_var\_neg As Variant

Public ab, xy, sum\_neg\_rej, neg\_rej, mean\_neg\_r\_v, mean\_neg\_rej, dif\_neg\_rej, sum\_dif\_rej As Variant

Public mse\_neg\_rej, rmse\_neg\_rej As Variant

Public rej, sum\_rej, var\_neg\_rej, dif\_var\_neg\_rej, sum\_var\_neg\_rej As Variant

Public mse\_var\_neg\_rej, rmse\_var\_neg\_rej, cd As Variant

Public rn\_5, rn\_51, ii As Variant

```
Public last_rn5, last_rn51 As Variant
Public kkk As Variant
Dim m5(0 To 5000), m51(0 To 5000), randomgamma(0 To 5000), xx(0 To 5000) As Variant
Dim negrandom(0 To 5000), negreject1(0 To 5000), negreject2(0 To 5000) As Variant
Dim nbr1(0 To 5000), nbr2(0 To 5000) As Variant
Private Sub Form_Activate()
    answer = 0
    i = 0
    m5(0) = 0
    m51(0) = 0
    a5 = 0
    a5_1 = 0
    k = 0
    k_1 = 0
    w = 0
    w_1 = 0
    expo = 0
    x_expo = 0
    sum_expo = 0
    random_gamma = 0
    sum_gamma = 0
    m_gamma = 0
    dif_x_mean = 0
    sum_for_var = 0
    var_gamma = 0
    standard_dev = 0
    estimator_alpha = 0
    estimator_beta = 0
    dif_est_alpha = 0
    dif_est_beta = 0
    sum_dif_est_alpha = 0
    sum_dif_est_beta = 0
    mse_alpha = 0
    mse_beta = 0
    rmse_alpha = 0
    rmse_beta = 0

    aa = 0
    bb = 0
    qq = 0
```

```
zeta = 0
dd = 0
vv = 0
yy = 0
zz = 0
ww = 0
xx(0) = 0
yy = 0
sum_xx = 0
m_gamma_acc_rej = 0
dif_xx_mean = 0
sum_for_var_acc_rej = 0
var_gamma_acc_rej = 0
standard_dev_acc_rej = 0
estimator_alpha_acc_rej = 0
estimator_beta_acc_rej = 0
dif_est_alpha_acc_rej = 0
dif_est_beta_acc_rej = 0
sum_dif_est_alpha_acc_rej = 0
sum_dif_est_beta_acc_rej = 0
mse_alpha_acc_rej = 0
mse_beta_acc_rej = 0
rmse_alpha_acc_rej = 0
rmse_beta = 0
est_mean_gamma = 0
est_variance_gamma = 0
sum_est_mean_gamma = 0
sum_est_variance_gamma = 0
est_mean_gamma_acc_rej = 0
est_variance_gamma_acc_rej = 0
sum_est_mean_gamma_acc_rej = 0
sum_est_variance_gamma_acc_rej = 0
mse_mean_gamma = 0
rmse_mean_gamma = 0
mse_variance_gamma = 0
rmse_variance_gamma = 0
mse_mean_gamma_acc_rej = 0
rmse_mean_gamma_acc_rej = 0
mse_variance_gamma_acc_rej = 0
rmse_variance_gamma_acc_rej = 0
```

```
geo_random = 0
neg_random = 0
p_estimator = 0
dif_p = 0
dif_pp = 0
sum_dif_p = 0
dif_neg = 0
sum_for_var_neg = 0
var_neg = 0
mse_var_neg = 0
rmse_var_neg = 0
mse_neg = 0
rmse_neg = 0
negrandom(0) = 0
randomgamma(0) = 0
negreject1(0) = 0
negreject2(0) = 0
neg_rej = 0
mse_neg_rej = 0
rmse_neg_rej = 0
sum_dif_rej = 0
dif_neg_rej = 0
sum_neg_r_v = 0
sum_neg_rej = 0
mean_neg_r_v = 0
mean_neg_rej = 0
```

```
rej = 0
sum_rej = 0
sum_var_neg_rej = 0
mse_var_neg_rej = 0
rmse_var_neg_rej = 0
cd = 0
```

Randomize Timer

loop\_test = InputBox("Loop Test ")

select\_define\_value = InputBox("Define value = 1 , Random Value = 2")

If select\_define\_value = 1 Then

```
sample = InputBox("For Gamma : Sample Test ")
alpha = InputBox("For Gamma : Alpha ")
```

```
beta = InputBox("For Gamma : Beta ")
```

```
number_trial = InputBox("Negative Binomial : number_trial ")
x_success = InputBox("Xth of success ")
prob_success = InputBox("Probability of Xth success ")
```

```
Else
```

```
Randomize Timer
```

```
sample = Int(Rnd * 1000)
```

```
alpha = Int(Rnd * 100)
```

```
beta = Round(Rnd, 4)
```

```
number_trial = Int(Rnd * 1000)
```

```
x_success = Int(Rnd * 100)
```

```
If x_success >= number_trial Or x_success = 0 Or sample = 0 Or alpha = 0 Or beta = 0 Then
```

```
Do While x_success >= number_trial Or x_success = 0 Or sample = 0 Or alpha = 0 Or beta  
= 0
```

```
Randomize Timer
```

```
sample = Int(Rnd * 1000)
```

```
alpha = Int(Rnd * 100)
```

```
beta = Round(Rnd, 4)
```

```
number_trial = Int(Rnd * 1000)
```

```
x_success = Int(Rnd * 100)
```

```
Loop
```

```
End If
```

```
prob_success = Round(Rnd, 4)
```

```
End If
```

```
For j = 1 To loop_test
```

```
answer = 0
```

```
answer_1 = 0
```

```
m5(0) = 0
```

```
a5 = 0
```

```
k = 0
expo = 0
x_expo = 0
sum_expo = 0
random_gamma = 0
sum_gamma = 0
m_gamma = 0
dif_x_mean = 0
sum_for_var = 0
var_gamma = 0
standard_dev = 0
estimator_alpha = 0
estimator_beta = 0
dif_est_alpha = 0
dif_est_beta = 0

m51(0) = 0
a5_1 = 0
k_1 = 0
xx(0) = 0
sum_xx = 0
m_gamma_acc_rej = 0
dif_xx_mean = 0
sum_for_var_acc_rej = 0
var_gamma_acc_rej = 0
standard_dev_acc_rej = 0
estimator_alpha_acc_rej = 0
estimator_beta_acc_rej = 0
dif_est_alpha_acc_rej = 0
dif_est_beta_acc_rej = 0

est_mean_gamma = 0
est_variance_gamma = 0

est_mean_gamma_acc_rej = 0
est_variance_gamma_acc_rej = 0

aa = 0
bb = 0
qq = 0
```

```
zeta = 0  
dd = 0  
vv = 0  
yy = 0  
zz = 0  
ww = 0  
xx(0) = 0  
yy = 0
```

```
dif_p = 0  
dif_pp = 0  
neg_rej = 0  
sum_neg_r_v = 0  
sum_neg_rej = 0  
mean_neg_r_v = 0  
mean_neg_rej = 0
```

```
negreject1(0) = 0  
negreject2(0) = 0  
dif_for_var_neg = 0  
dif_var_neg_rej = 0  
dif_neg_rej = 0  
cd = 0
```

```
For i = 1 To sample  
    Call crn_m5  
    Call gamma_inverse_t_c  
    Call gamma_acc_rej  
    Call nb  
Next i
```

```
For i = 1 To sample  
    Call nb_rej  
    Call find_nb
```

```
Next i  
Call mean_gamma  
Call var_and_std_gamma  
Call estimate_alpha_beta  
Call difference_est
```



Call mean\_negative

Call sum\_dif\_est\_p

Next j

Call mse\_rmse\_gamma

Call mse\_rmse\_neg

End Sub

Function crn\_m5()

Randomize Timer

m5(0) = Round(Rnd \* 10000)

a5 = Round(Rnd \* 10000)

k = Round(Rnd \* 10000)

w = Round(Rnd \* 10000)

m51(0) = Round(Rnd \* 10000)

a5\_1 = Round(Rnd \* 10000)

k\_1 = Round(Rnd \* 10000)

w\_1 = Round(Rnd \* 10000)

If m5(0) Or a5 Or k Or w = 0 Then

Do Until m5(0) And a5 And k And w <> 0

Randomize Timer

m5(0) = Round(Rnd \* 10000)

a5 = Round(Rnd \* 10000)

k = Round(Rnd \* 10000)

w = Round(Rnd \* 10000)

Loop

End If

ii = i - 1

last\_rn5 = m5(ii)

rn\_5 = ((a5 \* last\_rn5) + k) Mod w

answer = rn\_5 / w

m5(i) = rn\_5

If answer = 0 Then

Do Until answer <> 0

Randomize Timer

```
m5(0) = Round(Rnd * 10000)
a5 = Round(Rnd * 10000)
k = Round(Rnd * 10000)
w = Round(Rnd * 10000)
```

```
If m5(0) Or a5 Or k Or w = 0 Then
  Do Until m5(0) And a5 And k And w <> 0
    Randomize Timer
    m5(0) = Round(Rnd * 10000)
    a5 = Round(Rnd * 10000)
    k = Round(Rnd * 10000)
    w = Round(Rnd * 10000)
```

```
  Loop
End If
```

```
rn_5 = ((a5 * last_rn5) + k) Mod w
answer = rn_5 / w
m5(i) = rn_5
```

```
  Loop
End If
```

```
If m51(0) Or a5_1 Or k_1 Or w_1 = 0 Then
  Do Until m51(0) And a5_1 And k_1 And w_1 <> 0
    Randomize Timer
    m51(0) = Round(Rnd * 10000)
    a5_1 = Round(Rnd * 10000)
    k_1 = Round(Rnd * 10000)
    w_1 = Round(Rnd * 10000)
```

```
  Loop
End If
```

```
ii = i - 1
last_rn51 = m51(ii)
rn_51 = ((a5_1 * last_rn51) + k_1) Mod w_1
answer_1 = rn_51 / w_1
m51(i) = rn_51
```

```
If answer_1 = 0 Then
  Do Until answer_1 <> 0
    Randomize Timer
    m51(0) = Round(Rnd * 10000)
    a5_1 = Round(Rnd * 10000)
    k_1 = Round(Rnd * 10000)
    w_1 = Round(Rnd * 10000)

    If m51(0) Or a5_1 Or k_1 Or w_1 = 0 Then
      Do Until m51(0) And a5_1 And k_1 And w_1 <> 0
        Randomize Timer
        m51(0) = Round(Rnd * 10000)
        a5_1 = Round(Rnd * 10000)
        k_1 = Round(Rnd * 10000)
        w_1 = Round(Rnd * 10000)
      Loop
    End If
    rn_51 = ((a5_1 * last_rn51) + k_1) Mod w_1
    answer_1 = rn_51 / w_1
    m51(i) = rn_51
  Loop
End If

End Function

Function gamma_inverse_t_c()
  For a = 1 To alpha
    expo = -(beta) * Log(answer)
    x_expo = Round(expo, 4)
    sum_expo = sum_expo + x_expo
  Next a
  randomgamma(i) = sum_expo
  sum_gamma = sum_gamma + randomgamma(i)
End Function

Function gamma_acc_rej()
  aa = 1 / Sqr(2 * alpha - 1)
  bb = alpha - Log(4)
  qq = alpha + (1 / aa)
  zeta = 4.5
  dd = 1 + Log(zeta)
  vv = aa * Log(answer / (1 - answer))
  yy = alpha * (2.718) ^ vv
```

```
zz = answer ^ 2 * answer_1
ww = bb + (qq * vv) - yy
If ww + dd - (zeta * zz) >= 0 Then
    xx(i) = yy
Else
    If ww >= Log(zz) Then

        xx(i) = yy
    Else

        Do Until w >= Log(zz)
            Call crn_m5
            vv = aa * Log(answer / (1 - answer))
            yy = alpha * (2.718) ^ vv
            zz = answer ^ 2 * answer_1
            ww = bb + (qq * vv) - yy
        Loop
        xx(i) = yy

    End If
End If

sum_xx = sum_xx + xx(i)
End Function
Function mean_gamma()
    m_gamma = sum_gamma / sample
    m_gamma_acc_rej = sum_xx / sample
End Function
Function var_and_std_gamma()
    For i = 1 To sample
        dif_x_mean = (randomgamma(i) - m_gamma) ^ 2
        sum_for_var = sum_for_var + dif_x_mean

        dif_xx_mean = (xx(i) - m_gamma_acc_rej) ^ 2
        sum_for_var_acc_rej = sum_for_var_acc_rej + dif_xx_mean
    Next i
    var_gamma = sum_for_var / (sample - 1)
    standard_dev = Sqr(var_gamma)
    var_gamma_acc_rej = sum_for_var_acc_rej / (sample - 1)
    standard_dev_acc_rej = Sqr(var_gamma_acc_rej)
```

End Function

Function estimate\_alpha\_beta()

```
estimator_alpha = (m_gamma / standard_dev) ^ 2
estimator_beta = var_gamma / m_gamma
estimator_alpha_acc_rej = (m_gamma_acc_rej / standard_dev_acc_rej) ^ 2
estimator_beta_acc_rej = var_gamma_acc_rej / m_gamma_acc_rej
```

End Function

Function difference\_est()

```
dif_est_alpha = (estimator_alpha - alpha) ^ 2
sum_dif_est_alpha = sum_dif_est_alpha + dif_est_alpha
dif_est_beta = (estimator_beta - beta) ^ 2
sum_dif_est_beta = sum_dif_est_beta + dif_est_beta
dif_est_alpha_acc_rej = (estimator_alpha_acc_rej - alpha) ^ 2
sum_dif_est_alpha_acc_rej = sum_dif_est_alpha_acc_rej + dif_est_alpha_acc_rej
dif_est_beta_acc_rej = (estimator_beta_acc_rej - beta) ^ 2
sum_dif_est_beta_acc_rej = sum_dif_est_beta_acc_rej + dif_est_beta_acc_rej
est_mean_gamma = (m_gamma - (alpha * beta)) ^ 2
est_variance_gamma = (var_gamma - (alpha * beta ^ 2)) ^ 2
sum_est_mean_gamma = sum_est_mean_gamma + est_mean_gamma
sum_est_variance_gamma = sum_est_variance_gamma + est_variance_gamma
est_mean_gamma_acc_rej = (m_gamma_acc_rej - (alpha * beta)) ^ 2
est_variance_gamma_acc_rej = (var_gamma_acc_rej - (alpha * beta ^ 2)) ^ 2
sum_est_mean_gamma_acc_rej = sum_est_mean_gamma_acc_rej +
est_mean_gamma_acc_rej
sum_est_variance_gamma_acc_rej = sum_est_variance_gamma_acc_rej +
est_variance_gamma_acc_rej
```

End Function

Function mse\_rmse\_gamma()

```
mse_alpha = sum_dif_est_alpha / loop_test
rmse_alpha = Round(Sqr(mse_alpha), 4)
mse_beta = sum_dif_est_beta / loop_test
rmse_beta = Round(Sqr(mse_beta), 4)
mse_alpha_acc_rej = sum_dif_est_alpha_acc_rej / loop_test
rmse_alpha_acc_rej = Round(Sqr(mse_alpha_acc_rej), 4)
```

```
mse_beta_acc_rej = sum_dif_est_beta_acc_rej / loop_test
rmse_beta_acc_rej = Round(Sqr(mse_beta_acc_rej), 4)
mse_mean_gamma = sum_est_mean_gamma / loop_test
rmse_mean_gamma = Round(Sqr(mse_mean_gamma), 4)
mse_variance_gamma = sum_est_variance_gamma / loop_test
rmse_variance_gamma = Round(Sqr(mse_variance_gamma), 4)
mse_mean_gamma_acc_rej = sum_est_mean_gamma_acc_rej / loop_test
rmse_mean_gamma_acc_rej = Round(Sqr(mse_mean_gamma_acc_rej), 4)
mse_variance_gamma_acc_rej = sum_est_variance_gamma_acc_rej / loop_test
rmse_variance_gamma_acc_rej = Round(Sqr(mse_variance_gamma_acc_rej), 4)
```

```
Print "Loop Test = " & loop_test, "Sample test= " & sample
Print ""
Print "Alpha= " & alpha, "Beta= " & beta
Print "Inverse Transform and Convolution method for Gamma Distribution"
```

```
Print "RMSE estimate alpha = " & rmse_alpha, "RMSE estimate beta = " & rmse_beta
Print "RMSE estimate mean gamma = " & rmse_mean_gamma, "RMSE estimate variance = " &
rmse_variance_gamma
```

```
Print ""
```

```
Print "Accept and Reject method for Gamma Distribution"
Print "RMSE estimate alpha = " & rmse_alpha_acc_rej, "RMSE estimate beta = " &
rmse_beta_acc_rej
Print "RMSE estimate mean gamma = " & rmse_mean_gamma_acc_rej, "RMSE variance = " &
rmse_variance_gamma_acc_rej
Print ""
```

```
End Function
```

```
Function nb()
```

```
    For aa = 1 To x_success
        geo_random = (Log(answer) / Log(1 - prob_success)) - 1
        neg_random = neg_random + geo_random
        neg_r_v = Round(neg_random, 4)
    Next aa
    negrandom(i) = neg_r_v
    sum_neg_r_v = sum_neg_r_v + negrandom(i)
```

```
End Function
```

```
Function nb_rej()
  For ab = 1 To number_trial
    Randomize Timer
    Call crn_m5
    negreject1(ab) = answer
  Next ab
End Function
Function find_nb()
  For ab = 1 To number_trial
    If negreject1(ab) < p Then
      nbr1(ab) = negreject1(ab)
    Else
      nbr2(ab) = negreject1(ab)
    End If
    If negreject1(ab) < p And ab = s Then
      negreject2(i) = nbr2(ab)
    End If
  Next ab
  sum_neg_rej = sum_neg_rej + negreject2(i)
End Function
Function mean_negative()
  mean_neg_for_estimate = sum_neg_r_v / sample
  mean_neg_rej = sum_neg_rej / sample
  mean_estimate = x_success / probab_success
  For i = 1 To sample
    dif_neg = (negrandom(i) - mean_neg_for_estimate) ^ 2
    sum_for_var_neg = sum_for_var_neg + dif_neg
    rej = (negreject2(i) - mean_neg_rej) ^ 2
    sum_rej = sum_rej + rej
  Next i
  var_neg = sum_for_var_neg / (sample - 1)
  var_neg_rej = sum_rej / (sample - 1)
End Function
Function sum_dif_est_p()
  dif_p = (mean_neg_for_estimate - mean_estimate) ^ 2
  sum_dif_p = sum_dif_p + dif_p
  dif_neg_rej = (mean_neg_rej - mean_estimate) ^ 2
  sum_dif_rej = sum_dif_rej + dif_neg_rej
```

End Function

Function mse\_rmse\_neg()

    mse\_neg = sum\_dif\_p / loop\_test

    rmse\_neg = Round(Sqr(mse\_neg), 4)

    mse\_neg\_rej = sum\_dif\_rej / loop\_test

    rmse\_neg\_rej = Round(Sqr(mse\_neg\_rej), 4)

Print "Loop Test= " & loop\_test, "Sample= " & sample, "Number of Trial= " & number\_trial

Print "Xth Success= " & x\_success, "Probability of success= " & prob\_success

Print ""

Print "Inverse Transform and Convolution method for Negative Binomial Distribution"

Print "RMSE estimate negative binomial estimator by mean = " & rmse\_neg

Print ""

Print "The Acceptance And Rejection method for Negative Binomial Distribution"

Print "RMSE estimate negative binomial estimator by mean = " & rmse\_neg\_rej

End Function