**Electronic Supplementary Information(ESI)**

**Design of controlled-morphology NiCo2O4 with tunable and excellent microwave absorption performance**

Xiaogang Sua, Jun Wanga,b,\*, Xiaoxiao Zhanga, Siqi Huod, Wei Chenf, Wei Daia, and Bin Zhangc,\*

a School of Materials Science and Engineering, Wuhan University of Technology, Wuhan 430070, People’s Republic of China

b Institute of Advanced Material Manufacturing Equipment and Technology, Wuhan University of Technology, Wuhan 430070, People’s Republic of China

c School of Materials Science and Engineering, Wuhan Textile University, Wuhan 430200, People’s Republic of China

d School of Materials Science and Engineering, Wuhan Institute of Technology, Wuhan 430205, People’s Republic of China

f School of Civil Engineering, Architecture and Environment, Hubei University of Technology, Wuhan 430068, People’s Republic of China

Correspondence to: Jun Wang (E-mail: wgdfrp@whut.edu.cn)

Bin Zhang (E-mail: whutfrpzhangbin@163.com)

**EDX**

**Fig. S1.** EDX image of NC-F1 with atomic percentages

**介电损耗副本**

**Fig. S2.** Tangent value (tan δε, tan δm) of dielectric loss (a) and magnetic loss (b)

C-C

**Fig. S3.** (a-d) ε″ versus ε′ of NC-RS, NC-F1, NC-F2, and NC-F3 (Cole-Cole plots)

**拟合推导电导率**

**Fig. S4.** The fitting conductivity σ dependent on frequency of NC-RS, NC-F1, NC-F2, and NC-F3

**Appendix:**

Non-linear least squares fitting by running python is used to fit the values of conductive loss and polarization. According to this theory, the model function in this work is *ε=ε∞+(εs-ε∞)/(1+iωτ)-iσ/(ωε0)*, where *ε∞* is optical dielectric constant, *εs* is static dielectric constant, *ε0* is free space dielectric constant, *τ* is the relaxation time and *σ* is the conductive ratio. The four parameters of εs, ε∞, σ and τ signed as a group *β are* fitted . The sum of squares is , where *ri* is *εfit-ε*. Then adjusting *β* optimizes and obtain theminimum valus of S. The details code is below written.

*import numpy as np*

*from scipy.optimize import least\_squares*

*import xlrd*

*def realimag(array):*

*return np.array([(x.real, -x.imag) for x in array])*

*def func(x,p):*

*s,u,sigma,t=p #s:static dielectric constant;u:optical dielectric*

*constant;sigma:conductive ratio;t:relaxation time*

*d=complex(0,1)*

*o=8.854187817\*10\*\*(-12)*

*return realimag( u+(s-u)/(1+x\*x\*\*t\*t)*

*-np.dot(d,np.dot(np.dot(x,t),np.divide((s-u),(1+np.dot(np.dot(np.dot(x,x),t),t)))))*

*-np.dot(d,np.divide(sigma,np.dot(x,o))) )*

*def condut\_loss\_result(x,p):*

*s,u,sigma,t=p*

*o=8.854187817\*10\*\*(-12)*

*return np.divide(sigma,np.dot(x,o))*

*def residuals(p,y,x):*

*return (realimag(np.array(y)) - func(x, p)).flatten()*

*p0 = [20,10,1,10\*\*-12]*

*data=xlrd.open\_workbook('original\_data.xlsx')*

*data\_number=200*

*groups=20*

*table=data.sheets()[3]*

*fcost=0*

*fsig=0 #fitting conductive ratio*

*ft=0 #fitting relaxation time*

*fplsq=[]*

*fs=0 #fitting static dielectric constant*

*fu=0 #fitting optical dielectric constant*

*fconduct\_loss=0 #fitting conducting loss*

*frelaxation\_loss=0 #fitting polarization loss*

*for i in range(1,int(groups)+1):*

*end=i\*(int(data\_number/groups))*

*start=end-int(data\_number/groups)*

*#print("Reading data group",i,": from ",start," to ",end)*

*xdata=table.col\_values(0)[start:end]*

*ydata\_1=table.col\_values(2)[start:end] #imaginary permitivity*

*ydata\_2=table.col\_values(1)[start:end] #real permitivity*

*ydata=[]*

*for m in range(int(data\_number/groups)):*

*xdata[m]=xdata[m]\*10\*\*9 # GHz*

*ydata.append(complex(ydata\_2[m],-ydata\_1[m]))*

*plsq = least\_squares(residuals, p0,*

*bounds=([0,0,0,0],[200,200,200,10\*\*-1]),args=(ydata, xdata),max\_nfev=100000)*

*fplsq.append(plsq)*

*print(plsq.x[0]," ",plsq.x[1]," ",plsq.x[2]," ",plsq.x[3]," ")*

*conduct\_loss=np.mean(condut\_loss\_result(xdata,plsq.x))*

*#print("Conduct\_loss ",i,":",conduct\_loss)*

*fconduct\_loss += conduct\_loss*

*relaxation\_loss=np.mean(ydata\_1)-conduct\_loss*

*#print("Relaxation\_loss",i,":",relaxation\_loss)*

*#print("Imaginary:",np.mean(ydata\_1),", Conduct Loss:",conduct\_loss,",*

*Relaxation Loss:",relaxation\_loss)*

*#print(np.mean(xdata)/10\*\*9," ",np.mean(ydata\_1)," ",conduct\_loss,"*

*",relaxation\_loss)*

*frelaxation\_loss += relaxation\_loss*

*fsig=fsig+plsq.x[2]*

*fs=fs+plsq.x[0]*

*fu=fu+plsq.x[1]*

*ft=ft+plsq.x[3]*

*fcost=plsq.cost+fcost*