

---

# Dicke Hamiltonian, Reduced Basis

## Jz, J+, J- Construction

```
In[54]:= Jz[K_Integer] := Module[{J, dim, Jz},
    J = K / 2;
    dim = 2 J + 1;
    Jz = SparseArray[DiagonalMatrix[Reverse[Range[-J, J]]]];
    Return[Jz];
]

Jp[K_Integer] := Module[{J, dim, Jplus, mValues, i, j},
    J = K / 2;
    dim = 2 J + 1;
    Jplus = SparseArray[{}, {dim, dim}];
    mValues = Reverse[Range[-J, J]];
    For[i = 1, i < dim, i++,
        Jplus[[i, i + 1]] = Sqrt[J * (J + 1) - mValues[[i + 1]] * (mValues[[i + 1]] + 1)];
    ];
    Return[SparseArray[Jplus]];
]

Jm[K_Integer] := Module[{J, dim, Jplus, mValues, i, j},
    J = K / 2;
    dim = 2 J + 1;
    Jplus = SparseArray[{}, {dim, dim}];
    mValues = Reverse[Range[-J, J]];
    For[i = 2, i ≤ dim, i++,
        Jplus[[i, i - 1]] = Sqrt[J * (J + 1) - mValues[[i]] * (mValues[[i]] + 1)];
    ];
    Return[Jplus];
]
```

## Complete Construction

```
In[118]:=
(*Constants*)
bsize = 30;  $\omega_0$  = 1.0;  $\omega_c$  = 1.0; j = 0.07; K = 3;
(*Identity matrix for QH0*)
idH0 = SparseArray[IdentityMatrix[bsize]];
idTLS = SparseArray[IdentityMatrix[K + 1]];

(*QH0 Hamiltonian*)
H0H0 =  $\omega_c$  * SparseArray[Band[{1, 1}]  $\rightarrow$  Table[ $n + \frac{1}{2}$ , {n, 0, bsize - 1}]]];

(*Combined TLS Hamiltonian*)
HTLS =  $\omega_0$  * Jz[K];
(*Annihilation operator definition*)
a = SparseArray[Band[{1, 2}]  $\rightarrow$  Table[Sqrt[n], {n, 1, bsize - 1}], {bsize, bsize}];

Hindep = KroneckerProduct[HTLS, idH0] + KroneckerProduct[idTLS, H0H0];
Hcoup = j * (KroneckerProduct[Jp[K], a] + KroneckerProduct[Jm[K], a†]);
Htot = Hindep + Hcoup;
```

---

## Initial States, Observables Construction

### Initial States

```
In[127]:=
(*QH0*)
 $\psi_0H0$  = SparseArray[{1  $\rightarrow$  1.0}, bsize];
(*TLS*)
 $\psi_0TLS$  = SparseArray[{K - 1  $\rightarrow$  1.0}, K + 1];
Print[ $\psi_0TLS$  // MatrixForm];
 $\psi_0vec$  = KroneckerProduct[ $\psi_0TLS$ ,  $\psi_0H0$ ] // Flatten;
Print[Norm[ $\psi_0vec$ ]];


$$\begin{pmatrix} 0 \\ 1. \\ 0 \\ 0 \end{pmatrix}$$

1.
```

## Observable Matrices

### Oscillator Position

In[132]:=

```
xM = KroneckerProduct[IdentityMatrix[K + 1],  $\frac{1}{\text{Sqrt}[2]} (a^\dagger + a)$ ];  
ConjugateTranspose[psi0vec].xM.psi0vec
```

Out[133]=

0.

---

## Propagation

### Oscillator Expected Position

In[134]:=

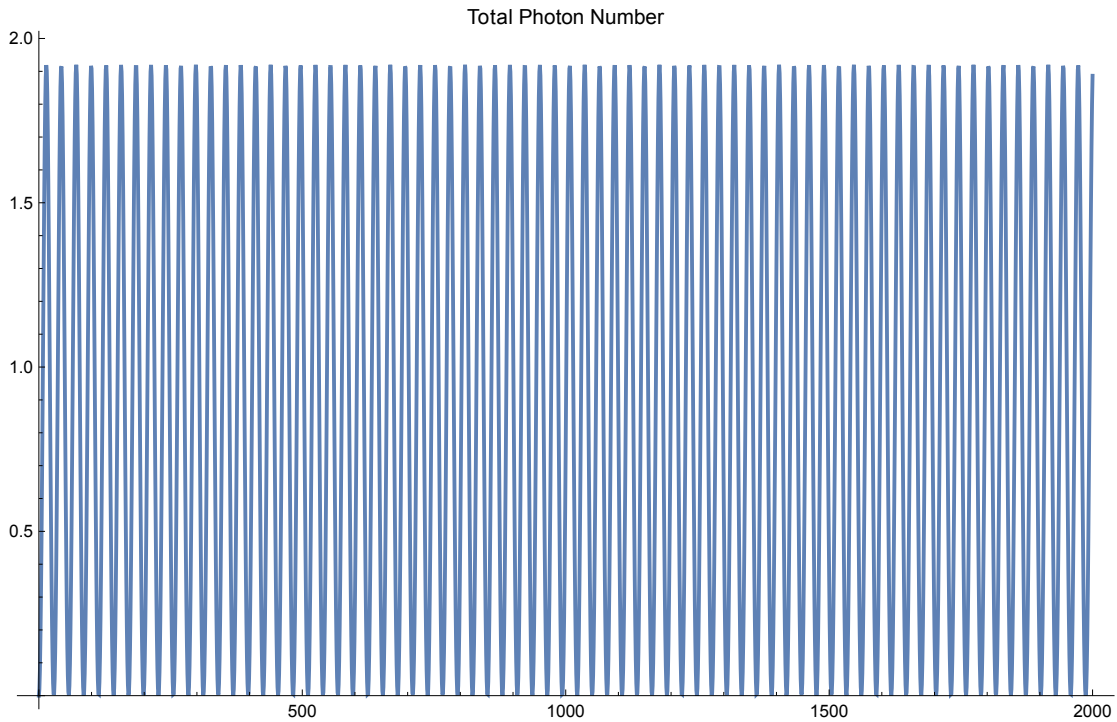
```
stateVector[t_] := MatrixExp[-I * Htot * t, psi0vec];  
tMax = 2000;  
tRange = Range[0, tMax, 1];  
psiS = ParallelTable[stateVector[t], {t, tRange}];
```

## Photon Number Expectation in Cavity

In[138]:=

```
aDaggerA = KroneckerProduct[IdentityMatrix[K + 1], a†.a];  
aDaggerAsr = aDaggerA.aDaggerA;  
photons = Table[Conjugate[ψs[[n]]].aDaggerA.ψs[[n]], {n, Length@tRange}];  
ListLinePlot[{tRange, photons // Re} // Transpose,  
  PlotRange → All, PlotLabel → "Total Photon Number", ImageSize → Large]
```

Out[141]=

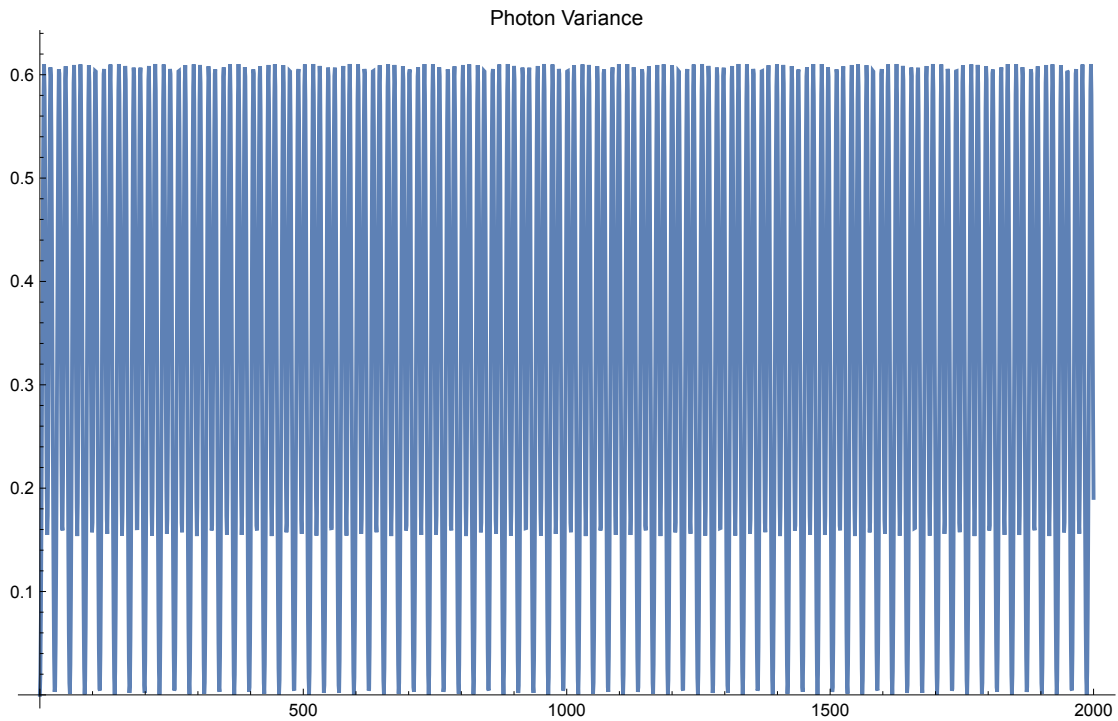


## Photon Statistics, Variance

In[142]:=

```
newPhotons =  
  Table[Conjugate[ψs[[n]].aDaggerAsr.ψs[[n]], {n, Length@tRange}] - photons^2;  
ListLinePlot[{tRange, newPhotons // Re} // Transpose,  
  PlotRange → All, PlotLabel → "Photon Variance", ImageSize → Large]
```

Out[143]=



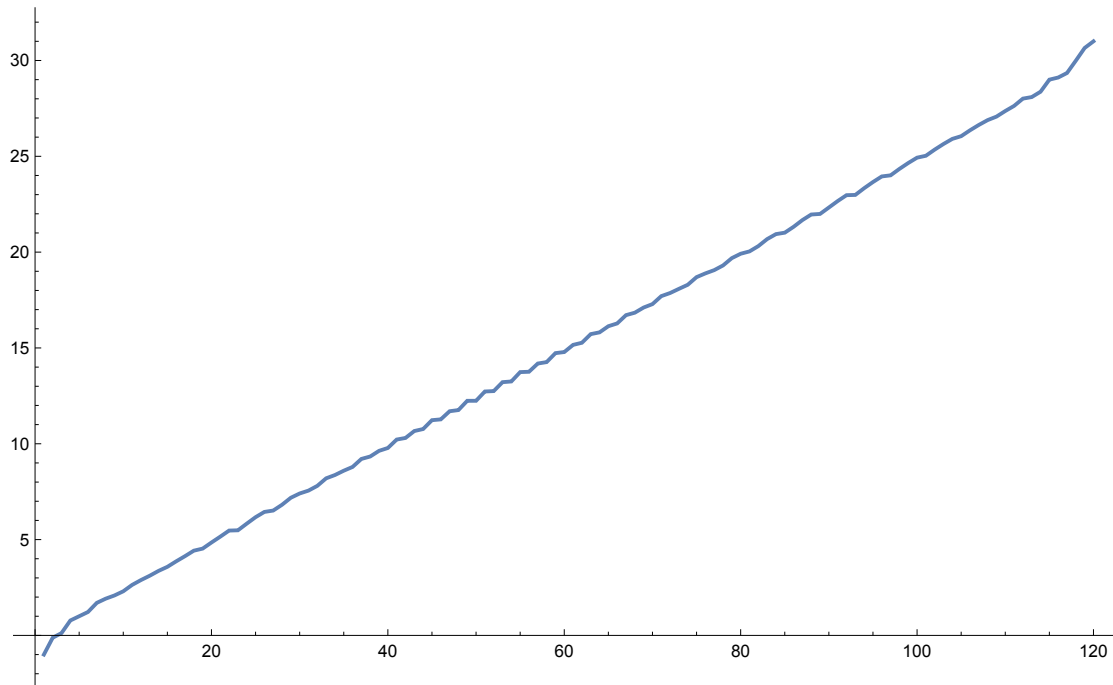
## Excitation Spectrum

In[144]:=

```
eigv = Eigenvalues[N[Htot]];  
ListLinePlot[{Sort[eigv]}, PlotRange → All, ImageSize → Large]
```

... **Eigenvalues** : Because finding 120 out of the 120 eigenvalues and /or eigenvectors is likely to be faster with dense matrix methods, the sparse input matrix will be converted. If fewer eigenvalues and /or eigenvectors would be sufficient, consider restricting this number using the second argument to Eigenvalues.

Out[145]=



In[ ]:=