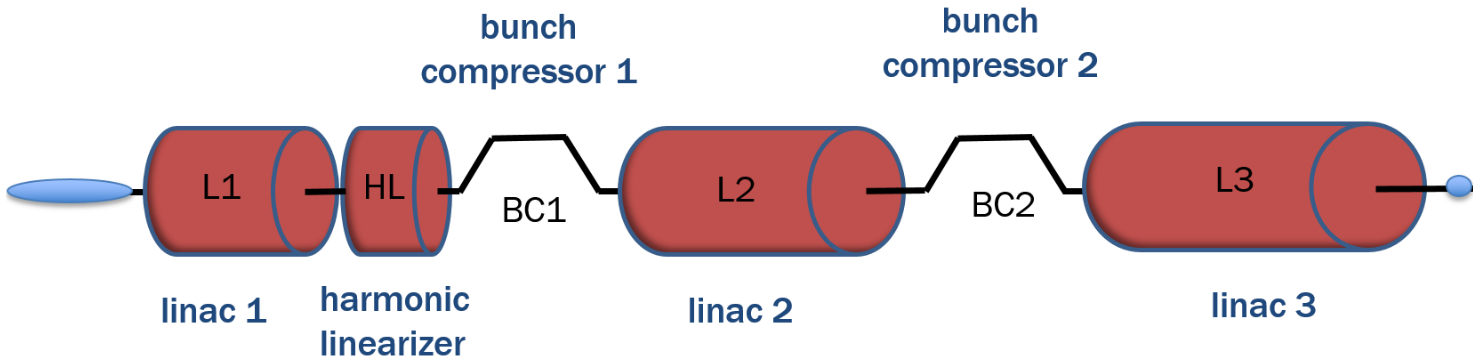
Practice Session Problem (cont'd)



- Initial beam conditions: 100pC, 100 MeV, total 3 mm beam with longitudinally uniform distribution in z, Gaussian energy distribution with standard deviation 2keV, no chirp, transverse round upright 4D Gaussian distribution with 0.2 mm RMS size, and 0.5 um normalized emittance.

- Linac parameters: L1 (1.3GHz, 20m, 16MV/m), k = 0.61685
  HL (3.9GHz, 5m, 10MV/m), k = 0.61685
  L2 (1.3GHz, 200m, 16MV/m), k = 0.61685
  L3 (1.3GHz, 400m, 16MV/m), k = 0.61685
  BC1 R56 = 5 cm, L = 10m, k = 0.27416
  BC2 R56 = 5 cm, L = 20m, k = 0.27416

- Undulator parameters: undulator period: 2cm, strength (K): 1.5.

Practice Session Problem

1) Find the final electron beam bunch length at the end of L3, estimate FEL radiation wavelength, bandwidth, and power with initial initial emittance, compute speedup of your parallel program

2) Find the final electron beam energy spread at the end of L3, estimate FEL radiation wavelength, bandwidth, and power with initial emittance, compute speedup of your parallel program

3) Find the final electron beam emittance at the end of L3, estimate FEL radiation wavelength, bandwidth, and power with initial peak current and maximum final energy, compute speedup of your parallel program

Sampling of Distribution

- Sample the Gaussian probability density function:

$$\phi'(y|0,1) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}y^2\right], \quad -\infty < y < \infty,$$

- Form a 2D Gaussian probability density function:

$$f(y_1, y_2) = \phi'(y_1|0,1)\,\phi'(y_2|0,1) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}\left(y_1^2 + y_2^2\right)\right].$$

- Change the coordinate:

$$Y_1 = R\cos\Phi,$$

$$Y_2 = R\sin\Phi,$$

$$\phi'(y_1)\phi'(y_2)\,\mathrm{d}y_1\,\mathrm{d}y_2 = \left(\exp\left[-\frac{1}{2}r^2\right] r\,\mathrm{d}r\right)\left(\frac{1}{2\pi}\,\mathrm{d}\phi\right).$$

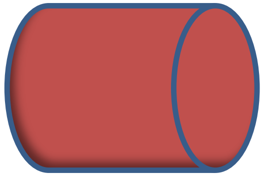
$$\Phi = 2\pi\xi_2.$$

$$Y_1 = [-2\log\xi_1]^{\frac{1}{2}}\cos 2\pi\xi_2,$$

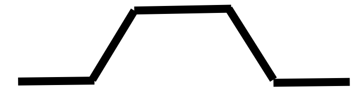
$$Y_2 = [-2\log\xi_1]^{\frac{1}{2}}\sin 2\pi\xi_2;$$

Longitudinal Dynamics Inside the Accelerator for Problem 1 and 2

RF cavity



bunch compressor



z: longitudinal relative position deviation w.r.p. reference particle position

$\Delta\gamma$: longitudinal normalized relative energy deviation w.r.p. reference particle energy

$$z^+ = z_1 + \frac{L_{\rm acc}}{2}\Delta\gamma_1/(\gamma_{01}\beta_{01})^3$$

$$\gamma_0^+ = \gamma_{01} + \frac{L_{\rm acc}}{2}\frac{qV_{\rm acc}}{mc^2}\cos(\phi_0)$$

$$\Delta\gamma_2 = \Delta\gamma_1 + L_{\rm acc}\frac{qV_{\rm acc}}{mc^2}(\cos(\phi_0 - kz^+) - \cos(\phi_0))$$

$$z_2 = z^+ + \frac{L_{\rm acc}}{2}\Delta\gamma_2/(\gamma_0^+\beta_0^+)^3$$

$$\gamma_{02} = \gamma_0^+ + \frac{L_{\rm acc}}{2}\frac{qV_{\rm acc}}{mc^2}\cos(\phi_0)$$

$$z = z + R_{56}\frac{\Delta\gamma}{\gamma_0} + T_{566}\left(\frac{\Delta\gamma}{\gamma_0}\right)^2 + U_{5666}\left(\frac{\Delta\gamma}{\gamma_0}\right)^3$$

$$\Delta\gamma_2 = \Delta\gamma_1$$

$$R_{56} \approx 2\theta^2\left(L_{db} + \frac{2}{3}L_b\right)$$

$$T_{566} \approx -\frac{3}{2}R_{56}$$

$$U_{5666} \approx 2R_{56}$$

Transfer Matrix through Linac and Bunch Compressor for Problem 3

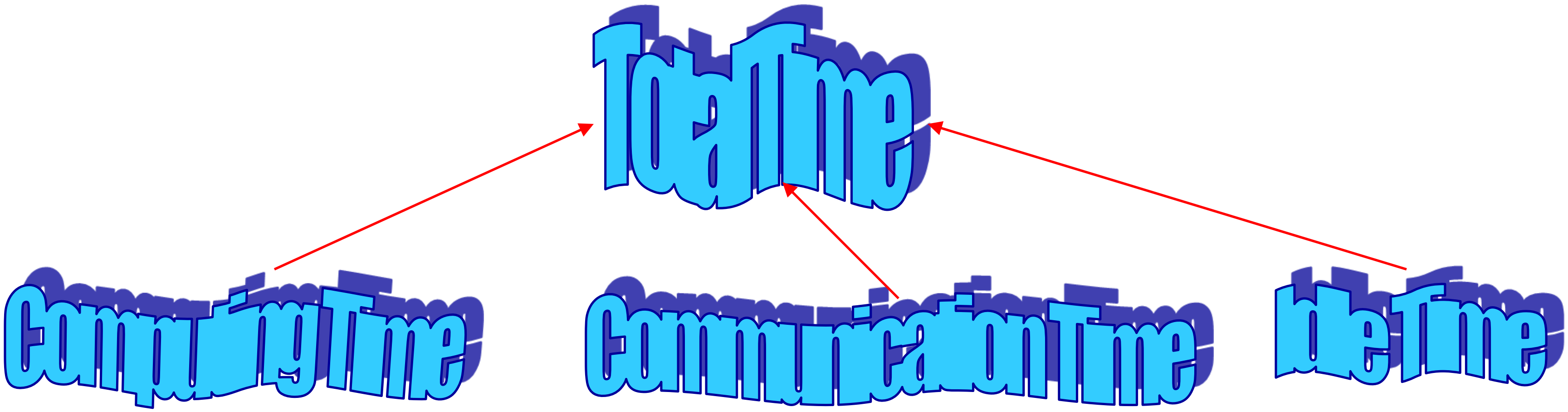
Linac Transfer Matrix

$$
\left(\begin{array}{cc|cc|cc}
\cos\sqrt{k}z & \frac{1}{\sqrt{k}}\sin\sqrt{k}z & & & & \\
-\sqrt{k}\sin kz & \cos\sqrt{k}z & & & & \\
\hline
 & & \cos\sqrt{k}z & \frac{1}{\sqrt{k}}\sin\sqrt{k}z & & \\
 & & -\sqrt{k}\sin kz & \cos\sqrt{k}z & & \\
\hline
 & & & & 1 & \frac{1}{\gamma_o^2\beta_o^2}z \\
 & & & & 0 & 1
\end{array}\right)
$$

Bunch Compressor Transfer Matrix

$$
\left(\begin{array}{cc|cc|cc}
\cos\sqrt{k}z & \frac{1}{\sqrt{k}}\sin\sqrt{k}z & & & & \\
-\sqrt{k}\sin kz & \cos\sqrt{k}z & & & & \\
\hline
 & & \cos\sqrt{k}z & \frac{1}{\sqrt{k}}\sin\sqrt{k}z & & \\
 & & -\sqrt{k}\sin kz & \cos\sqrt{k}z & & \\
\hline
 & & & & 1 & \text{R56} \\
 & & & & 0 & 1
\end{array}\right)
$$

Parallel Implementation



parallel speedup

$$S(n,P) = \frac{T(n,1)}{T(n,P)}$$

parallel efficiency

$$E(n,P) = \frac{S(n,P)}{P} = \frac{T(n,1)}{PT(n,P)}$$

$$T = (1 - \alpha)\, T + \alpha\, T$$

α is the fraction of time that cannot be done in parallel

$T(P) = (1 - \alpha)\, T/P + \alpha\, T$ $\quad S(P) = 1/[(1 - \alpha)\, 1/P + \alpha]$ $\quad S(\infty) = 1/\alpha$

❖ **the smaller fraction of serial time, the better parallel efficiency!!**

MPI Code Example

C Language - Blocking Message Passing Routines Example

```c
#include "mpi.h"
#include <stdio.h>

int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, dest, source, rc, count, tag=1;
char inmsg, outmsg='x';
MPI_Status Stat;

MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

if (rank == 0) {
  dest = 1;
  source = 1;
  rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
  }

else if (rank == 1) {
  dest = 0;
  source = 0;
  rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
  rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  }

rc = MPI_Get_count(&Stat, MPI_CHAR, &count);
printf("Task %d: Received %d char(s) from task %d with tag %d \n",
       rank, count, Stat.MPI_SOURCE, Stat.MPI_TAG);


MPI_Finalize();
}
```

Fortran - Blocking Message Passing Routines Example

```fortran
   program ping
   include 'mpif.h'

   integer numtasks, rank, dest, source, count, tag, ierr
   integer stat(MPI_STATUS_SIZE)
   character inmsg, outmsg
   outmsg = 'x'
   tag = 1

   call MPI_INIT(ierr)
   call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
   call MPI_COMM_SIZE(MPI_COMM_WORLD, numtasks, ierr)

   if (rank .eq. 0) then
      dest = 1
      source = 1
      call MPI_SEND(outmsg, 1, MPI_CHARACTER, dest, tag,
 &            MPI_COMM_WORLD, ierr)
      call MPI_RECV(inmsg, 1, MPI_CHARACTER, source, tag,
 &            MPI_COMM_WORLD, stat, ierr)

   else if (rank .eq. 1) then
      dest = 0
      source = 0
      call MPI_RECV(inmsg, 1, MPI_CHARACTER, source, tag,
 &        MPI_COMM_WORLD, stat, err)
      call MPI_SEND(outmsg, 1, MPI_CHARACTER, dest, tag,
 &        MPI_COMM_WORLD, err)
   endif

   call MPI_GET_COUNT(stat, MPI_CHARACTER, count, ierr)
   print *, 'Task ',rank,': Received', count, 'char(s) from task',
  &         stat(MPI_SOURCE), 'with tag',stat(MPI_TAG)

   call MPI_FINALIZE(ierr)

   end
```