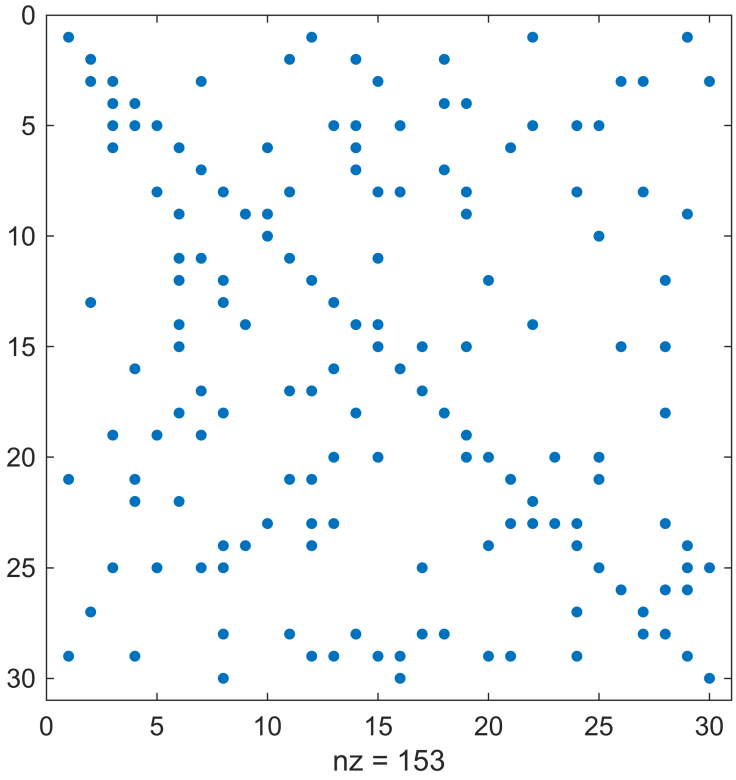
```
clear
rng('default')
```

Create a random matrix to analyse.

```
n = 30;
density = 0.15;
A = speye(n,n) + sprand(n, n, density);
spy(A) % the "spy plot" exhibits the locations of the nonzero entries
```



(We'll just check that the eigenvector matrix is well conditioned.)

```
[X,Lambda] = eig(full(A)); cond(X)
```

```
ans =
144.8732
```

Here is the full set of eigenvalues of A.

```
lambda = eig(full(A))
```

```
lambda = 30×1 complex
   3.1141 + 0.0000i
   2.2001 + 0.0000i
   0.1530 + 0.6561i
   0.1530 - 0.6561i
   1.0730 + 0.9468i
```

```
   1.0730 - 0.9468i
   0.1459 + 0.0000i
   1.5979 + 0.6602i
   1.5979 - 0.6602i
   1.6923 + 0.3273i
     ⋮
     ⋮
```

Now we run the Arnoldi method up to $m = 8$.

```
m = 8;
b = rand(n,1);
[V, H, beta] = arnoldi(A, b, m); % a separate function that does the same thing
```

We can check all our favourite relations still hold:

```
Vm = V(:, 1:m); Hm = H(1:m, 1:m);
norm(A*Vm - V*H)
```

```
ans =
3.0067e-16
```

```
em = [zeros(m-1,1); 1];
norm(A*Vm - Vm*Hm - H(m+1,m)*V(:,m+1)*em')
```

```
ans =
2.7538e-16
```

```
norm(Vm'*A*Vm - Hm)
```

```
ans =
3.4503e-15
```

Now we find the eigenvalues θ_i of H_m, compare with λ_i:

```
[Y, theta] = eig(H(1:m,1:m), 'vector'); theta, lambda
```

```
theta = 8×1 complex
   3.1140 + 0.0000i
   2.2224 + 0.0000i
   0.1146 + 0.6037i
   0.1146 - 0.6037i
   0.9818 + 0.7631i
   0.9818 - 0.7631i
   1.2351 + 0.4538i
   1.2351 - 0.4538i
lambda = 30×1 complex
   3.1141 + 0.0000i
   2.2001 + 0.0000i
   0.1530 + 0.6561i
   0.1530 - 0.6561i
   1.0730 + 0.9468i
   1.0730 - 0.9468i
   0.1459 + 0.0000i
   1.5979 + 0.6602i
   1.5979 - 0.6602i
```

```
   1.6923 + 0.3273i
     ⋮
```

The first seems mostly converged, and a couple of others seem to be getting there.

We can compute the residuals in two ways. First, we could actually form the Ritz vectors and subtract at the n-dimensional level.

```
U = V(:, 1:m) * Y  % Ritz vectors
```

```
U = 30×8 complex
  -0.0918 + 0.0000i  -0.0347 + 0.0000i   0.0264 + 0.0804i   0.0264 - 0.0804i ···
  -0.2502 + 0.0000i   0.2801 + 0.0000i  -0.1096 + 0.0603i  -0.1096 - 0.0603i
  -0.1250 + 0.0000i   0.0591 + 0.0000i   0.0550 - 0.1153i   0.0550 + 0.1153i
  -0.1428 + 0.0000i  -0.1203 + 0.0000i   0.0711 + 0.0416i   0.0711 - 0.0416i
  -0.3174 + 0.0000i  -0.2934 + 0.0000i  -0.1296 + 0.1014i  -0.1296 - 0.1014i
  -0.1462 + 0.0000i   0.2075 + 0.0000i  -0.1832 + 0.3130i  -0.1832 - 0.3130i
  -0.0666 + 0.0000i   0.0164 + 0.0000i  -0.0647 + 0.0478i  -0.0647 - 0.0478i
  -0.2896 + 0.0000i  -0.4576 + 0.0000i   0.0472 - 0.0127i   0.0472 + 0.0127i
  -0.1720 + 0.0000i   0.0498 + 0.0000i   0.3347 - 0.0308i   0.3347 + 0.0308i
  -0.1120 + 0.0000i  -0.2137 + 0.0000i  -0.0248 + 0.0133i  -0.0248 - 0.0133i
     ⋮
```

```
sqrt(sum(abs(A*U - U*diag(theta)).^2))  % residuals
```

```
ans = 1×8
    0.0086    0.2601    0.3662    0.3662    0.6395    0.6395    0.7547    0.7547
```

But our theory also gives us a formula for the residuals at the small, m-dimensional scale: $|h_{m+1,m}|\,|y_m|$. Let's compare:

```
residuals = abs(H(m+1,m)) * abs(Y(m,:))
```

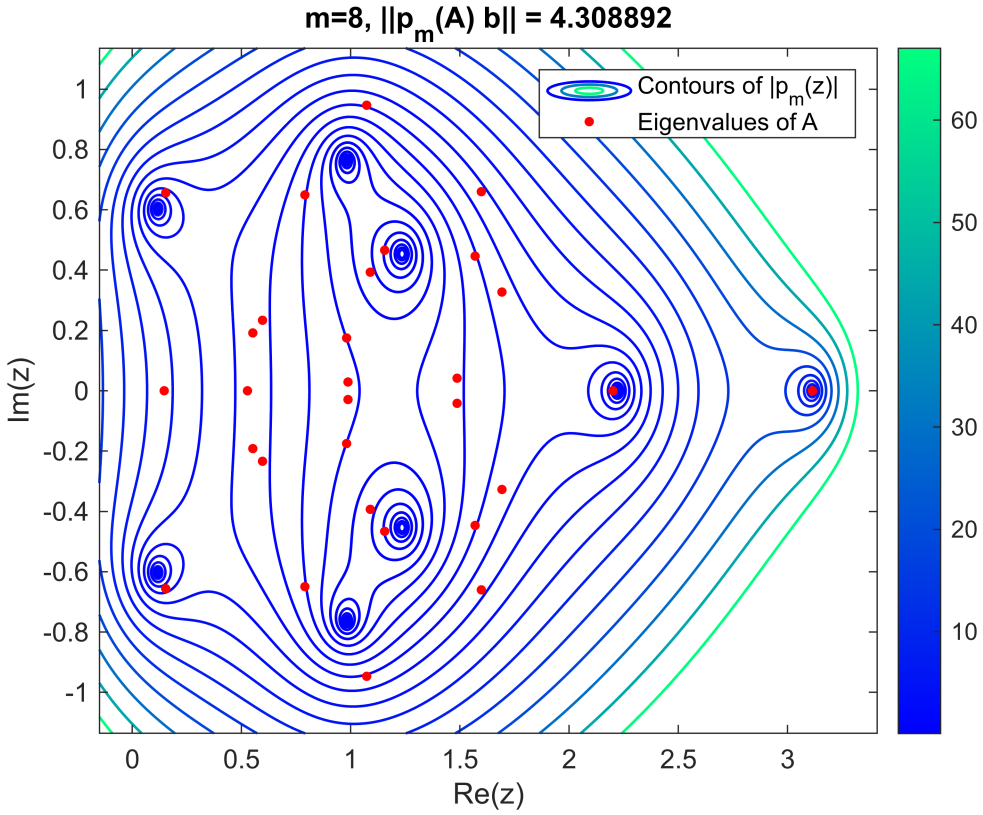
```
residuals = 1×8
    0.0086    0.2601    0.3662    0.3662    0.6395    0.6395    0.7547    0.7547
```

Confirmed.

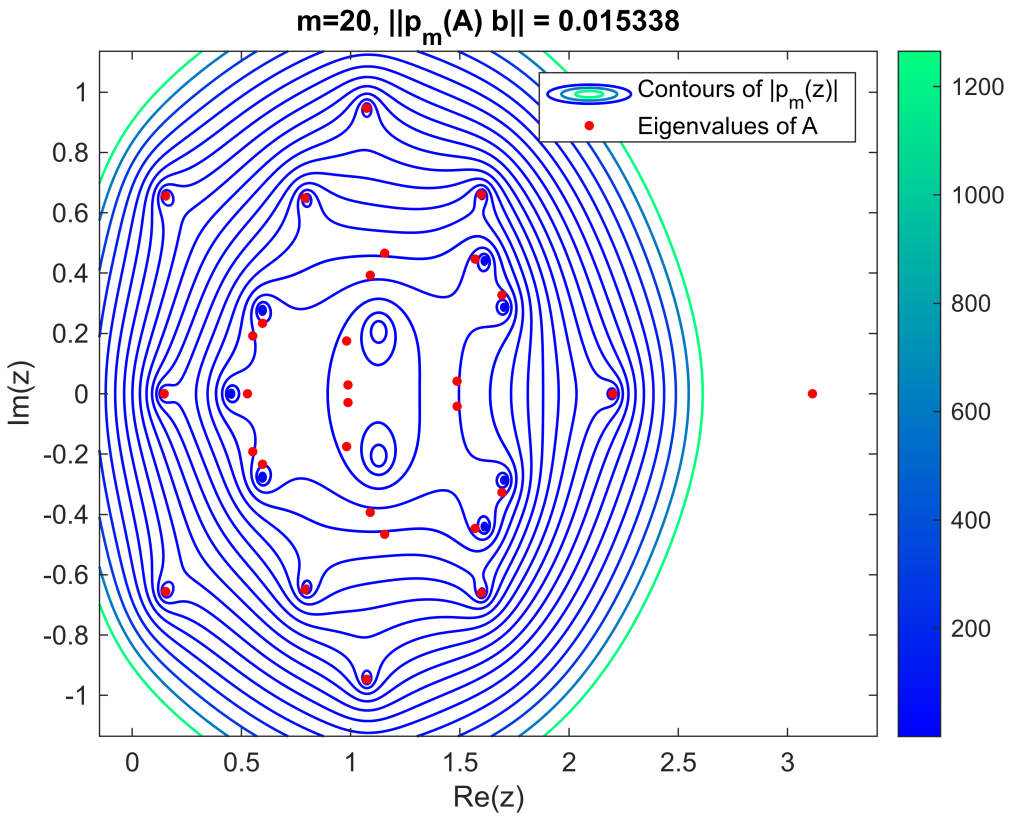
So now, the question is which eigenvalues does Arnoldi prioritise getting accurate estimates for? We will see it all comes down to a particular polynomial optimisation problem:

$$\text{minimise } \|p_m(A)b\| \text{ over all monic polynomials of degree } m.$$

```
figure
plot_arnoldi_polynomial(A, b, m)
```




```
figure
for mi = 1:20
    plot_arnoldi_polynomial(A, b, mi)
    drawnow
end
```


m=20, ||p_m(A) b|| = 0.015338
Contours of |p_m(z)|
Eigenvalues of A
Im(z)
Re(z)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
3
1200
1000
800
600
400
200